Volume 1
Issue 3

Jan/Feb
1988

Price £1.20

RISC USER

```
0008700  0D 00 0A 19 F4 20 20 20  .....
0008710  3E 4D 65 6D 45 64 69 74  >MemEdit
0008720  72 6F 67 72 61 6D 20 20  rogram
0008730  45 64 69 74 6F 72 0D 00  Editor..
0008740  69 6F 6E 20 20 20 41 20  ion   A
0008750  20 41 75 74 68 6F 72 20   Author
0008760  20 43 68 72 69 73 74 69   Christi
0008770  49 53 43 20 55 73 65 72  ISC User
00
00
00087A0  6
00087B0  50 1B
00087C0  34 2C 31 3
00087D0  20 F2 65 72 7
00087E0  69 74 69 61 6C
00087F0  6F 72 79 65 64 69
0008800  82 05 3A 0D 00 8C 1
0008810  65 64 69 74 6F 72 0D
0008820  46 58 20 32 31 2C 30
0008830  A5 0D 00 B4 0D C8 8E
         BE 18 C9 20 20 20 30
```

Risc User Memory Editor

THE MAGAZINE AND SUPPORT GROUP
EXCLUSIVELY FOR USERS OF THE ARCHIMEDES

Volume 1 Issue 3
January/February 1988

RISC USER

CONTENTS

RISC User is published by BEEBUG Ltd.
Co-Editors: Mike Williams, Dr Lee Calcraft
Assistant Editor: Kristina Lucas
Production Assistant: Yolanda Turuelo
Subscriptions: Mandy Mileham

BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts AL1 1EX. Tel. St.Albans (0727) 40303

Printed by Arlon House

The Archimedes Magazine
and Support Group.

EDITORIAL

We hope that we have now begun to establish a recogniseable 'RISC user' flavour, both with regard to style and content. We are grateful to those who have written to us already with their opinions on the magazine, but we will continue to listen carefully to the views and comments of all RISC User members who communicate with us. We will also be delighted to receive programs and articles for potential publication and we do pay for all material used. To help would-be contributors we have prepared a set of notes to provide some guidance on style and other matters. To receive a copy please ask for "RISC User Contributors' Notes" and send us an A5 SAE. The address for all communications appears at the foot of the contents page opposite.

As reported on the news page following, Acorn did not manage to send out more than a handful of Series One OS ROMs before Christmas, and it now appears likely that many users will still be without the new ROM at the time you receive this copy of RISC User. Several programs and articles in this issue were prepared on the basis that all readers would by this time have Series One installed in their machines. In view of the likely situation, we have added additional markers at the head of articles where appropriate showing where use of the Series One OS is essential (and in the case of the article on Fonts, the Series One Welcome Disc is required as well). We very much hope that Series One will be the standard for everybody by the time of the next issue of RISC User, and it is our firm intention to concentrate on Series One only as soon as possible. Do make sure you have returned your owner registration form to Acorn as this is the only way to obtain the upgrade.

We have established the monthly disc as an ideal companion to your copy of RISC User. To make this even better value, we are endeavouring to add additional items on to each disc, and so far all three discs have been filled to near capacity. On the disc for issue one we were able to include Computer Concepts' stunning animation of Newton's cradle, issue two contained a number of equally impressive screen displays from Clares, showing the use of their Artisan and Graphic Writer packages. For this month's disc we have included some of the best entries in our Painting Competition, the results of which are given on the next page. You will be able to load these into the Welcome paint package to make further modifications if you so wish. See the back cover for details.

Please note that this issue of RISC User covers both January and February (we publish ten times a year). The March issue should be with you near the beginning of that month.

NewsNewsNewsNewsNew

A round-up of the latest news and comment in the Archimedes world compiled by Mike Williams. All prices quoted below include VAT.

ACORN

The new managing director of Acorn replacing Brian Long who resigned suddenly last year, is Harvey Coleman of Olivetti (Acorn's majority shareholder). We very much hope that this will remove some of the current uncertainty at Acorn and that better times now lie ahead.

We reported in the last issue that Acorn expected to start shipping the 1.2 OS ROM before Christmas to existing owners of the Archimedes. Despite the fact that supplies were in the main warehouse, apparently very few were sent out in December and the latest indication is that the new ROM will be sent out during January, to be followed shortly after by ArcWriter (see article in this issue). We stress again that you must have returned the owner registration form that comes with your Archimedes to receive these free upgrades.

Whether Acorn has had second thoughts or not we don't know, but a spokesman for Acorn has stated that there is now no immediate intention to change the price of 'C', Fortran or Pascal as was suggested last month.

The offer of 0% finance spread over 12 months has proved very popular with potential purchasers, and Acorn has extended this to the 31st January. Moreover, this deal now covers the latest A440 as well as the A305, A310 and A310M. BEEBUG normally holds stocks of all these machines, and all the details required to purchase through this scheme can be given over the phone.

MORE ART FOR THE ARCHIMEDES

Another art package for the Archimedes has been released, this time by Fairhurst Instruments Ltd. It is claimed to be the first art program to use a 256 colour mode and is available at the special budget price of £19.95. The package is mouse-driven using icons, and while not offering all the facilities of Clares' Artisan (reviewed in the first issue of RISC User), does provide a worthwhile set of tools for the budding artist. Arctist is available from Fairhurst Instruments Ltd, Dean Court, Woodford Road, Wilmslow, Cheshire SK9 2LT or telephone (0625) 525694.

MORE SOFTWARE FROM CLARES

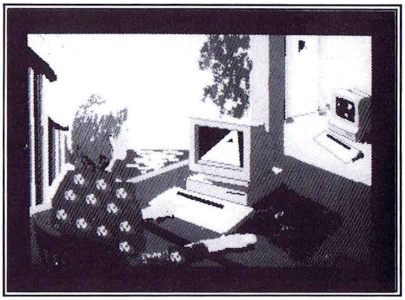
The full versions of Clares' Artisan and Graphic Writer (formerly called Image Writer), which were reviewed in the first and second issues of RISC User respectively, are now readily available at £39.95 and £29.95 (10% discount to RISC User members - see price list). Clares has said that a utilities disc for use with Artisan should be available by the end of January. This will provide an upgraded Integrex colour printer dump, dumps for a number of colour dot matrix printers, a number of slick fade routines, and probably some additional sprites and graphics. Details have yet to be finalised as has the price, but this should be under £20. Clares' promised database package, Alphabase, should also be out at about the same time (at a cost of £49.95), and we hope to be able to review this in the next issue of RISC User. For further details contact Clares Micro Supplies, 98 Middlewich Road, Rudheath, Northwich, Cheshire CW9 7DA or telephone (0606) 48511.

BBC MICROLIVE

We understand that the BBC are planning a special Microlive programme on graphics and animation in which the Archimedes will be heavily featured. There is no firm date yet for the transmission but look out around the end of January/beginning of February.

BEEBUG PAINTING COMPETITION

In the first issue of RISC User we invited readers to submit their best efforts using the Painting program on the Welcome disc. The winner, out of the entries received, has to be Kathie Lewis of Hampton Hill, Middlesex, but close runners up were



Elliott and Zoe Hughes (age 12 and 10 respectively) from Swanwick, Derbyshire, both of whom submitted individual pictures, and Jeroen Boomgaardt from Enschede in the Nerherlands who provided an excellent rendering of Acorn's Archimedes poster. Our thanks and congratulations to all who entered, and a prize of Artisan (thanks to Clares) has been sent to the winner. Clares has generously provided a second copy of Artisan and this is winging its way to Elliott and Zoe for their splendid efforts. We will be including the best of the pictures submitted on this month's magazine disc. RU

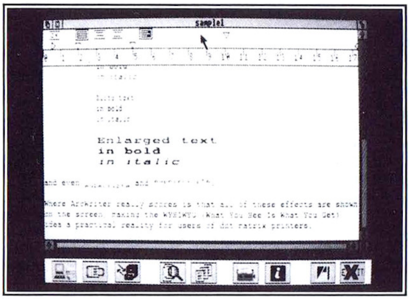
REVIEWS

ArcWriter

Mike Williams investigates ArcWriter, Acorn's free word processor for Archimedes users.

When the Archimedes was launched, Acorn promised that all early purchasers of the machine would be supplied eventually with a word processor free of charge. ArcWriter is the result, and Acorn says that it will continue to supply ArcWriter to all new owners (who register with Acorn) for the foreseeable future.

ArcWriter is a large (132K) Basic program which makes full use of the WIMP environment to provide a largely WYSIWYG style of word processor. There is one large window which contains your text, displayed as black on white, and which provides an overall limit to line width. Pressing the menu button on the mouse with the pointer in the text window displays the edit mode menu on screen. From here you can select fonts (Elite, pica, enlarged, subscript and superscript) and underline text, all displayed as such on the screen.



Elite text
in bold
in italic

Enlarged text
in bold
in italic

and even [illegible] and [illegible]

Where ArcWriter really scores is that all of these effects are shown on the screen, making the WYSIWYG (What You See Is What You Get) idea a practical reality for users of dot matrix printers.

Screen formatting is governed by a ruler measured in centimetres, and page margins, header and footer space is likewise specified in the same units. You can select paper size too (A4 and A5 in my version) but I believe other sizes will be possible). Tab stops are selected from a Tab 'well' and positioned on the ruler as required using the mouse pointer, a system that works well and is easy to understand. One bonus is that the bottom of each page shows up clearly on the screen as a grey line between lines of text. This is always useful information.

Text may be left, right or fully justified, or centred. These options are also selected from icons displayed with a ruler. In effect, every line of text visible on the screen has its own individual ruler. If the ruler is displayed on the screen then scrolling the ruler is displayed on the screen then scrolling through the text will show the ruler settings for each line in turn.

It is also feasible to mark blocks of text by moving the cursor to the start of a block, moving the pointer to the end of the block and pressing the adjust (right-hand mouse button). Once a block of text has been marked in this way it can be deleted, copied or moved elsewhere. In addition, ruler formatting and justification can be applied to the whole of the block.

The main menu bar across the foot of the screen provides nine different menus including an exit from ArcWriter. The other functions covered in this way are:

1. System Configuration - this covers choice of storage medium and printer.

2. Document Configuration - page layout, margins, headers, footers etc.

3. Filing Functions - saving, loading, deleting text files.

4. Search and Replace - this is case sensitive and cannot be changed.

5. Page Location - go to any specified page.

6. Printing Functions - allows first and last page for printing to be set.

7. Provision of Information - gives information about current document and use of system resources.

8. System Functions - this permits the use of star commands.

I have only been able to spend a limited time with ArcWriter, but I like what I see, and could readily see myself using this for many of my Archimedes word processing requirements. My main concern is with the visual quality of the display as characters have a definite tendency to appear blurred, but that is a function of the system not just ArcWriter. In addition, ArcWriter has a noticeable struggle to keep up when entering text at a reasonable (two finger) speed. Maybe ArcWriter doesn't provide all the features and sophistication of commercial word processors, but then it is all for free, at least to the user. RU

A PRINTER BUFFER FOR THE ARCHIMEDES.

By Nic Van Someren

O.S. 1.2 only

Use this robust Printer Buffer to free your computer during long printouts - whether of text files or programs.

The printer buffer described here assembles as a relocatable module which will provide a variable size printer buffer for your Archimedes. You can set it to just 100 bytes, or to half a megabyte if you wish. Once turned on, the buffer will operate automatically whenever the printer is engaged, and its contents will remain intact even if Escape or Break are pressed; though it is cleared on Ctrl- or Shift-Break and on Reset. It also has an emergency Clear facility, actioned by pressing the ALT key simultaneously with Escape.

RUNNING THE PROGRAM

First of all, type in the program carefully, and save it away. When it is run, if all goes well, it will create on disc a relocatable module file called BufModule. You can then load this by typing:

```
        *BUFMODULE
or      *RMLOAD BUFMODULE
```

To check that all is well, type:

```
        *MODULES
```

This should show that the module is present. Typing *HELP COM. should give further details. The module also responds to:

```
        *HELP BUFFER
and     *HELP BUFFERSIZE
```

As you will see from this, the buffer is switched on and off with *BUFFER ON and *BUFFER OFF (the space before the parameters "ON" and "OFF" is essential). You can find the current buffer size (default size 16K) by typing:

*BUFFERSIZE

This Printer Buffer program will form a part of a forthcoming book on the Arthur Operating System from Dabbs Press. We are most grateful to Nic Van Someren, the program author, and to Dabbs Press for permission to publish it here.

The allocation may be altered at any time that the buffer is in an OFF state by typing:

*BUFFERSIZE n

where n is the size in bytes. This is decimal by default, but may be preceded with "&" for hex, or even"2_" if you are giving the new size in binary.

The buffer appears to be extremely robust. There are just two provisos to note. Firstly, the contents of the buffer, but not the buffer itself, are lost under an *RMTIDY - but this is unlikely to be a problem. Secondly, because of what appears to be a bug in Arthur 1.2, you cannot extend the buffer size (using *BUFFERSIZE) if another relocatable module has been loaded above the buffer, or indeed if workspace for another module has been allocated above the buffer. In practice the problem is avoided by ensuring that the buffer is the **last** module to be loaded. So if you are using a !BOOT file to set up the RISC User Disc Menu and the Buffer, the relevant lines might appear as follows:

```
QUIT
*RMENU
*BUFMODULE
*BUFFERSIZE &8000       (Allocate 32K)
*BUFFER ON
*MENU                   (Call the Menu)
```

This program is featured on this month's magazine disc. For further details, see the back cover.

```
 10 REM              >BufSource4
 20 REM Program      Printer Buffer
 30 REM Requires     OS Series One
 40 REM Version      A 1.04
 50 REM Author       Nic Van Someren
 60 REM RISC User    Jan/Feb 1988
 70 REM Program      Subject to copyright
 80:
 90 DIM Q% 4000
100 FOR I=4 TO 7 STEP 3
```

```
  110 P%=0:O%=Q%
  120 [OPT I
  130;Module Header
  140 EQUD 0
  150 EQUD initialise
  160 EQUD finalise
  170 EQUD service
  180 EQUD title
  190 EQUD helpstring
  200 EQUD helptable
  210 EQUD 0;SWI chunk
  220 EQUD 0;SWI handler
  230 EQUD 0;SWI table
  240 EQUD 0;SWI code
  250.title
  260 EQUS "PrinterBuffer"
  270 EQUB 0
  280 ALIGN
  290.helpstring
  300 EQUS "Printer Buffer"+
CHR$9+"1.04 (16 Dec 1987)"
  310 EQUB 0
  320 ALIGN
  330.helptable
  340 EQUS "Buffer"
  350 EQUB 0
  360 ALIGN
  370 EQUD bufcommand
  380 EQUD &00010001;flags
  390 EQUD syntax
  400 EQUD bufhelp
  410 EQUS "BufferSize"
  420 EQUB 0
  430 ALIGN
  440 EQUD sizecommand
  450 EQUD &00010100
  460 EQUD sizesyntax
  470 EQUD sizehelp
  480 EQUD 0
  490.bufhelp
  500 EQUS "*Buffer turns bu
ffer ON or OFF"
  510 EQUB 13
  520 EQUB 10
  530.syntax
  540 EQUS "Syntax: *Buffer
<ON|OFF>"
  550 EQUB 0
  560.sizehelp
  570 EQUS "*BufferSize with
out a parameter gives the si
ze of the extra buffer"
  580 EQUB 13
  590 EQUS "With one paramet
er, the value is taken as th
e new buffer size"
  600 EQUB 13
  610.sizesyntax
  620 EQUS "Syntax: *BufferS
ize [<size>]"
  630 EQUB 0
  640 ALIGN
  650.bufcommand
  660 LDR R12,[R12]
  670 LDRB R2,[R0],#1
  680 ORR R2,R2,#&20
  690 CMP R2,#ASC"o"
  700 BNE badonoff
  710 LDRB R2,[R0],#1
  720 ORR R2,R2,#&20
  730 CMP R2,#ASC"n"
  740 BEQ setup
  750 CMP R2,#ASC"f"
  760 BNE badonoff
  770 LDRB R2,[R0],#1
  780 ORR R2,R2,#&20
  790 CMP R2,#ASC"f"
  800 BEQ setdown
  810.badonoff
  820 ADR R0,bufneedonoff
  830 ORR R14,R14,#1<<28;Set
V flag for error
  840 MOVS PC,R14
  850.bufneedonoff
  860 EQUD &00123456
  870 EQUS "*Buffer needs ON
or OFF after it"
  880 EQUB 0
  890 ALIGN
  900.sizecommand
  910 STMFD R13!,{R14}
  920 MOV R11,R12
  930 LDR R12,[R12]
  940 CMP R1,#0
  950 BEQ tellsize
  960 ADR R2,areweon
  970 LDR R2,[R2]
  980 CMP R2,#0
  990 BNE changewhileon
 1000 MOV R1,R0
 1010 MOV R0,#10
 1020 SWI "OS_ReadUnsigned"
 1030 ADD R2,R2,#17
 1040 LDR R10,[R12,#4]
 1050 MOV R9,R2
 1060 SUB R3,R2,R10
 1070 MOV R2,R12
 1080 MOV R0,#13
 1090 SWI "OS_Module"
 1100 CMP R0,#13
 1110 BNE osmodulebug
 1120 STR R2,[R11]
 1130 MOV R3,R9
 1140 STR R3,[R2,#4]
 1150 MOV R3,#&10
 1160 STR R3,[R2]
 1170 STR R3,[R2,#8]
 1180 STR R3,[R2,#12]
 1190 LDMFD R13!,{PC}
 1200.tellsize
 1210 SWI "OS_WriteS"
 1220 EQUS "The extra printe
r buffer is "
 1230 EQUB 0
 1240 ALIGN
 1250 LDR R0,[R12,#4]
 1260 SUB R0,R0,#17
 1270 ADR R1,numbuffer
 1280 MOV R2,#11
 1290 SWI "OS_ConvertCardina
l4"
 1300 SWI "OS_Write0"
 1310 SWI "OS_WriteS"
 1320 EQUS " bytes long."
 1330 EQUB 13
 1340 EQUB 10
 1350 EQUB 0
 1360 ALIGN
 1370 LDMFD R13!,{PC}
 1380.osmodulebug
 1390 LDMFD R13!,{R14}
 1400 ORR R14,R14,#1<<28
 1410 MOVS PC,R14
 1420.changewhileon
 1430 LDMFD R13!,{R14}
 1440 ADR R0,changetext
 1450 ORR R14,R14,#1<<28
 1460 MOVS PC,R14
 1470.changetext
 1480 EQUD &00123457
 1490 EQUS "Buffer must be O
FF before changing size"
 1500 EQUB 0
 1510 ALIGN
 1520.numbuffer
 1530 EQUD 0
 1540 EQUD 0
 1550 EQUD 0
 1560.initialise
 1570 STMFD R13!,{R14}
 1580 MOV R1,R12
 1590 LDR R12,[R12]
 1600 CMP R12,#0;Check if al
ready initialised
 1610 BNE startmeup
 1620 MOV R0,#6
 1630 MOV R3,#&4000; <<<  TH
IS IS THE DEFAULT SIZE (16K)
 1640 SWI "OS_Module"
 1650 STR R2,[R1]
 1660 MOV R12,R2
 1670 STR R3,[R12,#4]
 1680 MOV R3,#&10
 1690 STR R3,[R12]
 1700 STR R3,[R12,#8]
 1710 STR R3,[R12,#12]
 1720.startmeup
 1730 LDMFD R13!,{PC}
 1740.finalise
 1750 STMFD R13!,{R14}
 1760 LDR R12,[R12]
 1770 BL setdown
 1780 CMP R10,#0
 1790 MOV R0,#7
 1800 MOV R2,R12
 1810 SWINE "OS_Module"
 1820 LDMFD R13!,{PC}
 1830.service
 1840 CMP R1,#&27
 1850 MOVNE PC,R14
 1860 STMFD R13!,{R0,R1}
 1870 ADR R1,areweon
 1880 MOV R0,#0
 1890 STR R0,[R1]
 1900 LDMFD R13!,{R0,R1}
 1910 MOV PC,R14
 1920.areweon
 1930 EQUD 0
 1940;+0 is start of buffer
from 0
 1950;+4 is end of buffer fr
om 0
 1960;+8 is the point for in
sertion
 1970;+12 is the point for r
emoval
 1980.nextval;Increment R4
 1990 LDR R5,[R12,#4]
 2000 ADD R4,R4,#1
 2010 CMP R4,R5
 2020 LDREQ R4,[R12]
 2030 MOVS PC,R14
 2040.isempty;Empty test
 2050 LDR R4,[R12,#8]
 2060 LDR R5,[R12,#12]
 2070 CMP R4,R5
 2080 MOV PC,R14
 2090.isfull;Full test
 2100 STMFD R13!,{R14}
 2110 LDR R4,[R12,#8]
 2120 BL nextval
 2130 LDR R5,[R12,#12]
 2140 CMP R4,R5
 2150 LDMFD R13!,{PC}
 2160.topush;Byte to buff
 2170 STMFD R13!,{R14}
 2180 BL isfull
 2190 LDMEQFD R13!,{R14}
 2200 ORREQS PC,R14,#1<<29
 2210 LDR R4,[R12,#8]
 2220 STRB R0,[R12,R4]
 2230 BL nextval
 2240 STR R4,[R12,#8]
 2250 LDMFD R13!,{R14}
 2260 BICS PC,R14,#1<<29
 2270.topull;Byte from buff
 2280 STMFD R13!,{R14}
 2290 BL isempty
 2300 LDMEQFD R13!,{R14}
 2310 ORREQS PC,R14,#1<<29
 2320 LDR R4,[R12,#12]
 2330 LDRB R0,[R12,R4]
 2340 MOV R2,R0
 2350 TST R6,#1<<28
 2360 BLEQ nextval
 2370 STR R4,[R12,#12]
 2380 LDMFD R13!,{R14}
 2390 BICS PC,R14,#1<<29
 2400.toaltflush
 2410 STMFD R13!,{R0,R1,R2,R
14}
 2420 MOV R0,#&81
 2430 MOV R1,#&FD;ALT key
 2440 MOV R2,#&FF
 2450 SWI "OS_Byte"
 2460 CMP R1,#&FF
 2470 LDMFD R13!,{R0,R1,R2,R
14}
 2480 MOVNE PC,R14
 2490 LDR R4,[R12]
 2500 STR R4,[R12,#8]
 2510 STR R4,[R12,#12]
 2520 MOV PC,R14
 2530.tocountpurge
 2540 TST R6,#1<<28
 2550 BNE toaltflush
 2560 STMFD R13!,{R14}
 2570 TSTP R6,R6
 2580 LDRCS R4,[R12,#8]
 2590 BLCS nextval
 2600 LDRCS R5,[R12,#12]
 2610 LDRCC R5,[R12,#8]
 2620 LDRCC R4,[R12,#12]
 2630 SUBS R1,R5,R4
 2640 BHI posspace
 2650 LDR R4,[R12]
 2660 LDR R5,[R12,#4]
 2670 ADD R1,R1,R5
 2680 SUB R1,R1,R4
 2690.posspace
 2700 MOV R2,R1,LSR #8
 2710 LDMFD R13!,{PC}
 2720.myinsv
 2730 CMP R1,#3
 2740 MOVNES PC,R14
 2750 STMFD R13!,{R0,R1,R4,R
5}
 2760 BL topush
 2770 LDMFD R13!,{R0,R1,R4,R
5,PC}
 2780.myremv
 2790 STMFD R13!,{R6}
 2800 MOV R6,PC
 2810 CMP R1,#3
 2820 LDMNEFD R13!,{R6}
 2830 MOVNES PC,R14
 2840 STMFD R13!,{R1,R4,R5}
 2850 BL topull
 2860 LDMFD R13!,{R1,R4,R5,R
6,PC}
 2870.mycnpv
 2880 STMFD R13!,{R6}
 2890 MOV R6,PC
 2900 CMP R1,#3
 2910 LDMNEFD R13!,{R6}
 2920 MOVNES PC,R14
 2930 STMFD R13!,{R4,R5}
 2940 BL tocountpurge
 2950 LDMFD R13!,{R4,R5,R6,P
C}
 2960.setup
 2970 STMFD R13!,{R14}
 2980 ADR R1,areweon;Get add
ress of ON flag
 2990 LDR R0,[R1];Load flag
 3000 CMP R0,#0;Test if zero
 3010 LDMNEFD R13!,{PC};If n
ot, buffer is running
 3020 MVN R0,#0;Set flag
 3030 STR R0,[R1]
 3040 MOV R2,R12
 3050 LDR R0,[R12];Empty buf
 3060 STR R0,[R12,#8]
 3070 STR R0,[R12,#12]
 3080 MOV R0,#&14
 3090 ADR R1,myinsv
 3100 SWI "OS_Claim"
 3110 MOV R0,#&15
 3120 ADR R1,myremv
 3130 SWI "OS_Claim"
 3140 MOV R0,#&16
 3150 ADR R1,mycnpv
 3160 SWI "OS_Claim"
 3170 LDMFD R13!,{PC}
 3180.setdown
 3190 STMFD R13!,{R14}
 3200 MOV R2,R12
 3210 ADR R1,areweon;ON test
 3220 LDR R0,[R1]
 3230 CMP R0,#0
 3240 LDMEQFD R13!,{PC}
 3250 MOV R0,#0
 3260 STR R0,[R1]
 3270 MOV R0,#&14
 3280 ADR R1,myinsv
 3290 SWI "OS_Release"
 3300 MOV R0,#&15
 3310 ADR R1,myremv
 3320 SWI "OS_Release"
 3330 MOV R0,#&16
 3340 ADR R1,mycnpv
 3350 SWI "OS_Release"
 3360 LDMFD R13!,{PC}
 3370 ]:NEXT
 3380 OSCLI"SAVE BufModule "
+STR$~Q%+"+"+STR$~P%
 3390 OSCLI"SETTYPE BUFMODUL
E FFA"
```

RU

ARCHIMEDES SCREEN SUPER-SAVER

Lee Calcraft presents a fast, legal screen-save routine that takes the palette into account, and copes with differently configured systems.

Three programs are presented here. The first two are EXEC files, and should be entered without line numbers (use Wordwise, Twin or *BUILD). They should be saved with the names FASTSAVE and FASTLOAD in the Library directory of your disc, and should be given a file type of &FFE (*SETTYPE filename FFE). Now, if you type *FASTSAVE, a filename will be requested. Enter the name of the screen (previously saved using *SCREENSAVE). The EXEC file will load in the screen, and create two new files: one called Name+ (where Name is the original name of the screen) containing palette information, and the other, called Name=, containing the screen itself. You can now delete the original screen if you wish.

If you now type *FASTLOAD, and supply the screen name e.g. Name (no "+" or "=" needed), the mode and palette will automatically be set up, and the screen loaded - all in about one quarter of the time taken by a *SCREENLOAD command.

```
*| >FastSave
*|
MODE0:VDU21,26:DIM B% 30
B%=((B%+3)DIV4)*4:!B%=148
!(B%+4)=7:!(B%+8)=-1
VDU6:INPUT"Filename "file$:OS.("SCREEN
  L. "+file$):file$=LEFT$(file$,9):SYS
  &31,B%,B%+&10:Z%=!(B%+&10):L%=!(B%+
  &14):OFF:VDU24|:OS.("SCREENS."+file$
  +"+"):OS.("SAVE "+file$+"= "+STR$~Z%
  +" +"+STR$~L%):ON

*| >FastLoad
*|
MODE0:VDU21,26:DIM B% 30
B%=((B%+3)DIV4)*4;!B%=148
!(B%+4)=-1SYS
&31,B%,B%+&10:Z%=!(B%+&10)
VDU6:INPUT"Load Filename "file$:OS.("
  SCREEN.L."+file$+"+"):OS.("LOAD"+fi
  le$+"= "+STR$~Z%)
```

The third program sets up a slide show using the "FastLoad" principle. The screens must all have been subjected to the FastSave routine, and their names placed in the last line of the program in place of the four given. The word "END" is used as a terminator.

Notes:
All 3 routines use OS_ReadVDUVariables (&31) to read the screen base address, and the save routine also uses this call to obtain the screen size. The palette is saved by setting the graphics window to zero size, and using *SCREENSAVE to save a single graphics pixel together with mode and palette information. The screen itself is saved using a straight *SAVE.

```
   10 REM >FastShow
   20 REM Fast-Load Slide Show
   30 REM By Lee Calcraft
   40 :
   50 MODE 0:DIM B% 30:Z%=FNaddr
   60 REPEAT:RESTORE
   70 REPEAT
   80 READ file$
   90 IF file$<>"END" THEN PROCload(file
$):OFF
  100 A=INKEY(500)
  110 UNTIL file$="END"
  120 UNTIL FALSE
  130 :
  140 DEFFNaddr
  150 B%=((B%+3)DIV4)*4
  160 !B%=148:!(B%+4)=-1
  170 SYS &31,B%,B%+&10
  180 =!(B%+&10)
  190 :
  200 DEFPROCload(file$)
  210 OSCLI("SCREENL. "+file$+"+")
  220 OSCLI("LOAD "+file$+"= "+STR$~Z%)
  230 ENDPROC
  240 :
  250 DATA GARDEN,IMAGE1,IMAGE2,IMAGE3,E
ND
```

RU

REVIEWS

▸LOGiSTiX◂ for the Archimedes

Reviewed by Mike Williams

Logistix is the first new major application for the Archimedes marketed by Acorn, and thus deserves particular attention at this time. It provides a significant spreadsheet facility combined with database and time management functions (of which more later), all of which can form the basis for some impressive graphical displays.

Logistix is not a brand new product, having been available for the IBM PC (and its clones) for some time as well as for the Atari ST range. Logistix has been transferred onto the Archimedes (it was originally written in C) and is sold by Acorn under the Acornsoft brand name. Logistix may be the first applications product converted for the Archimedes from other machines, but you can be sure it won't be the last.

Logistix is an impressive product in many ways. It consists of two discs together with a hefty two-part manual, a multi-page reference card and a keystrip, all packaged in two of Acorn's now standard large-size packs. The manuals are, with one glaring failure, well written and produced. As for the software, insert the master disc, type in a single command and the Logistix screen appears.

Anyone at all familiar with Acorn's View Professional will immediately recognise the fundamental concepts of Logistix. The screen display (apart from the generated graphics) always presents a spreadsheet style of layout with rows and columns. It is on this basic 'worksheet' that the intricacies of spreadsheets, database handling, time management and even some word processing are played out.

Many potential users may well feel daunted at their first sight of Logistix. The software indulges in none of the colourful windows and icons beloved of other software packages produced for the Archimedes, and you might as well throw the mouse away altogether. What Logistix does offer is a powerful and highly integrated package that does not get in your way when solving problems.

The 'cell cursor' can be moved about the worksheet with the usual cursor keys, and the Page Up, Page Down, Home and End (or Copy) keys also perform their expected functions. The spreadsheet is an enormous 1024 columns by 2048 rows, so large that several different activities relating to the same task can be set up in different parts of the same worksheet. Twenty-seven of the worksheet rows are visible on the screen at any time, with four additional lines at the foot of the screen for user communication. These are the *status* line, the *prompt* line, the *help* line and the *entry* line. All input is clearly prompted with a single line of help information, while screenfuls of help text are available on virtually everything that Logistix can do.

SPREADSHEET

It is probably as a spreadsheet that Logistix will most often be used. Individual cells may contain text, values or expressions. Expressions, referencing other cells can use the basic arithmetic (+ - * / ^) and logical operators (= <> > < >= >=), and nearly 30 mathematical functions (trig, hyperbolic etc). Formulae are easily replicated, either by direct input of the appropriate cell references, or by moving the cell cursor to the same cell positions. I found no difficulty in setting up a spreadsheet for some of my own data, and quickly learnt how to manipulate this.

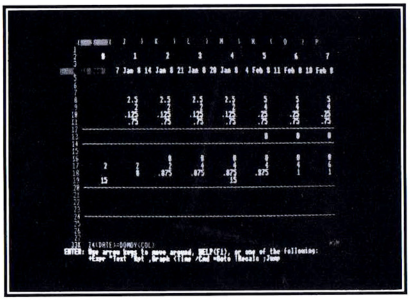
DATABASE

To operate Logistix as a database the columns of the spreadsheet layout are set up as fields (column widths can be varied by the user), and the rows correspond to individual records. Data of all kinds can be entered in the same way as for a spreadsheet, and by using cell references, it is very easy to link spreadsheet and database together. There are a number of specific database functions (for summation, counting, averaging etc) and the /T command provides sorting and searching functions. Records may only be sorted on one whole key field at a time, but successive sorts produce nested key sorts. With a maximum 64 fields and 2047 records, Logistix' capabilities as a database are clearly limited, but should be quite adequate for the context and expected use.

TIME MANAGEMENT

Logistix can handle the scheduling of resources (including people) over time, including critical path analysis. The software has a built in calendar that can be tailored to any particular task. For example, if you decide that a day is the basic unit of time appropriate to a job, you can specify which days in the calendar, and which hours per day are work time. Once created, the calendar is then linked to any selected column of the spreadsheet, and following days (if *day* is the basic time unit) occupy successive columns to the right. Resources can then be scheduled with the help of Logistix, and a critical path determined if this is appropriate.

There are many functions specific to time management, in fact too numerous to mention. In any case, scheduling a range of activities will take much less time than it would to describe. Again, by extensive use of cell references, the timesheet part of your worksheet can be easily linked to both spreadsheet and database applications. Thus a change to any parameter will propagate throughout ALL related cells.



```
        0       1        2        3        4        5        6        7
          7 Jan 0 14 Jan 0 21 Jan 0 28 Jan 0  4 Feb 0 11 Feb 0 18 Feb 0
Z4(DATE):=DOWDY(COL)
ENTER: Use arrow keys to move around, HELP(F1), or one of the following:
       +Expr "Text 'Opt ,Graph <Time /Cmd =Goto !Recalc ;Jump
```

GRAPHICS

The fourth prime function of Logistix is to allow all kinds of data held and generated in a worksheet to be displayed graphically. This uses a special graphics language to generate any of eight different types of chart (stacked bar charts, clustered bar charts, pie charts, line graphs, stepped graphs, spread and tick charts, scattergrams and Gantt charts - for work scheduling). Note, though, that the graphics produced are identical to those on the PC - none of the Archimedes' full range of colour and shade for example.

Using windows, up to four different charts may be shown on the same screen. Again, reference to cells (rather than to their contents) means that once a graph display has been programmed, any changes to the underlying values will automatically be reflected in the graphical representation. Graphs may also be output to a range of dot matrix printers (including the Epson range), and to a good many graph plotters. The graphics incorporate a number of fonts for stylish text, but I fear that legends and captions are not always that clear on the standard Archimedes colour screen.

MANUAL

In part one, the first 35 pages provide an excellent introduction to Logistix based around an example file supplied on disc. This gets across the fundamental features of spreadsheet, database, timesheet and graphics. BUT, and this is really an unforgiveable if understandable failure, the illustrations and the text all relate to a 20 row screen (standard for the PC) and thus all the cell references in the text are out of step with the screen display seen by the user (showing 27 rows)! In addition I noted four misprints, which could well confuse the less experienced user, and there could be more.

Apart from that, I thought that the two-part manual was clearly written and easy to follow, no mean feat with software as demanding as Logistix. The book also describes a further nine example Logistix files which are supplied on disc.

CONCLUSION

This abbreviated description of Logistix hardly does justice to its full potential, and much has been omitted, for example the facility to program macros. In reality, I suspect that it is only by using such software that you will come to appreciate its full value despite the absence of fancy windows, icons and the like.

Two major features stand out: the obvious power and speed that this software can provide on an Archimedes, and the way you can efficiently and directly create the applications you need. If you want a thoroughly businesslike environment for spreadsheet and timesheet applications then there is nothing to beat Logistix on an Archimedes, and the graphics are a real bonus. If you really want a sophisticated database (or word processing) package then look elsewhere. But Logistix does what it is designed to do superbly well. RU

ANTI-ALIAS FONTS FOR HUMANS

O.S. 1.2 only

Lee Calcraft describes how to use anti-alias fonts with a minimum of brain-strain.

The Tutorial and Demonstration files on the Welcome disc show the user what can be done using the special so-called "Fancy Font" facility on the Archimedes. A special Font Manager module resident in ROM is used to reproduce selected typefaces in a multitude of different sizes from font files supplied on the Welcome disc. The result is quite impressive, though if you try to use these fonts at a much greater point size, the letters begin to look a little moth-eaten. The Series One Welcome disc contains two sets of font files, one for a typeface called *Trinity* which looks similar to Times Bold, and another called *Corpus* which is reminiscent of the Courier typeface.

The way in which these fonts are used relies on a technique called "anti-aliasing". This essentially involves removing the jagged edges from displayed characters by replacing pixels which should be half on and half off, by ones of an intermediate shade. The demonstrations on the Welcome disc use 8 different shades of pixel to create their effect.

Of course, there is much more to using the Archimedes Fancy Fonts than anti-aliasing, and, as is obvious from the Reference Manual, their user interface is really quite complex. There are many SWI calls, and VDU23 statements, all with masses of parameters. What I propose to do, therefore, is to present in the first instance a short program which will display some text in Fancy Fonts. Then we will take a look at a couple of procedures which parcel up the Fancy Font calls in a more manageable way.

CONFIGURING FONT SIZE

Before making use of the machine's Fancy Fonts, you will need to allocate a block of RAM for storing font data. To do this, type:

```
    *CONFIG. FONTSIZE 10
```

Then press Ctrl-Break. This allocates 40K of font space (10 "pages" of 4K), which will be sufficient for experimenting with fonts of up to around 35 point. If you are using a 305, there should still be no problems with this setting, providing that your screen size is set to the default of 0 (i.e. 80K of screen RAM) - which is sufficient for running the two accompanying programs.

SETTING UP YOUR DISC

There are six stages to getting a Fancy Font onto the screen. The first involves setting up a disc with the correct font files. The remaining five steps may be executed in five lines of Basic, as we shall see in a moment. To prepare your disc, you will need to have on it a directory called FONTS. This must have a sub-directory with the name of the font which you will be using (either *Trinity* or *Corpus*). This directory must itself contain a further directory called *MEDIUM*. Finally MEDIUM must contain two files, called *IntMetrics* and *x90y45*. These contain scaling data for the fonts, and actual pixel data for each character, respectively. Put another way, if you are using the Trinity font, your disc should contain the two files:

```
    $.FONTS.TRINITY.MEDIUM.IntMetrics
    $.FONTS.TRINITY.MEDIUM.x90y45
```

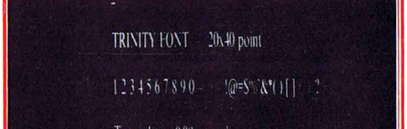
The simplest way to achieve this is to copy the whole directory called $.FONTS.TRINITY from the new Welcome disc, complete with its sub-directory and files. First create the directory $.FONTS on your work disc, then type:

```
    *COPY :0.$.FONTS.TRINITY :0.$.FONTS.TR
    INITY QPRF
```

and follow the prompts. The pair of files will use around 40K of disc space.

RUNNING THE PROGRAM

Once you have done this, you may run the program given in Listing 1. It should produce a display similar to that in the accompanying photograph. You will note that the time displayed is around 6 seconds. If you now run the program a second time, you should get a better result by a factor of 60 or more. This is because when the program is first run, the Font Manager must load the font data from disc, and store it in the so-called font cache. The second time around, there is no need for this. As you will appreciate, this is a very creditable result from a speed point of view. It is certainly fast enough for use within a word processor, as Acorn have demonstrated with ArcWriter.



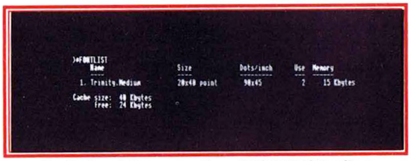
```
TRINITY FONT - 20x40 point

1234567890-   !@=$%&*()[]  ?

Time taken  0.09 seconds
```

1. Output from listing 1 (when RUN for the first time - subsequent runs will give a faster time displayed)

The Font Manager provides a special command, *FONTLIST for checking the state of the font cache. If you issue this now, you should see a display similar to that in the accompanying screen shot. This lists Trinity 20x40 point (i.e. Trinity font with horizontal point size 20, and vertical point size 40) as the only currently cached font. It reveals that the cache has 40K allocated to it, and that 24K of this is still free. We will return to this display later in the article. But now we will take a look at the five lines of Basic referred to earlier.



```
>*FONTLIST
    Name                  Size          Dots/inch    Use  Memory
    ----                  ----          ---------    ---  ------
 1. Trinity.Medium        20x40 point    90x45         2   15 Kbytes

Cache size:  40 Kbytes
      free:  24 Kbytes
```

2. Output from typing *FONTLIST after running prog 1.

Listing 1

```
   10 REM >Fonttest6
   20 REM Anti-Alias Font Demonstration
   30 REM By Lee Calcraft
   40 REM Calls Trinity font files
   50 :
   60 MODE12:TIME=0
   70 :
   80 REM Set Font Directory
   90 *SET Font$Prefix $.FONTS
  100 :
  110 REM Set Transfer Function
  120 VDU23,25,3,2,4,6,8,10,12,14
  130 :
  140 REM Set Anti-Aliasing Palette
  150 VDU 23,25,&88,9,0,0,0,&F0,&F0,&F0
  160 :
  170 REM Cache required Font
  180 SYS "Font_FindFont",,"Trinity.Medi
um",20*16,40*16,0,0
  190 :
  200 S$="TRINITY FONT  -  20x40 point"
  210 REM Paint Font
  220 SYS "Font_Paint",,S$,&10,0,800
  230 :
  240 S$="1 2 3 4 5 6 7 8 9 0 + = - !@#$
%^&*( ) [ ] < > ?"
  250 REM Paint Font
  260 SYS "Font_Paint",,S$,&10,0,600
  270 :
  280 S$="Time taken = "+STR$(TIME/100)+
" seconds"
  290 SYS "Font_Paint",,S$,&10,0,400
```

WHAT THE PROGRAM DOES

The first key statement in the program given in listing 1 appears at line 90. Its function is just to tell the Font Manager where to find the relevant font files. In this particular case, the command informs the Font Manager that the TRINITY directory, with its nested contents is located in $.FONTS. The next instruction, at line 120, sets up the so-called Transfer Function for the anti-aliasing. It determines how many gradations will be used in the anti-aliasing palette, and at what levels of brightness these shades will be used. The user can select 2, 4, 8 or 16 levels, the latter requiring a 256 colour mode. In our example we have used 8 levels of shading. If you are using 4 levels only, you could replace this line by VDU23,25,2,4,8,12|. For further details of this rather nasty VDU call, the reader is referred to the Reference Manual.

ANTI-ALIASING PALETTE

The next call, made in line 150, shares its first two parameters (23,25) with the Transfer Function. Its syntax is:

```
VDU23,25,bcg,startcol,0,0,rs,gs,bs,
re,ge,be
```

The variable bcg specifies the background colour number, and must have &80 added to it. In our example, we are using colour 8 for the background, so the bcg parameter is &80 + 8. The next parameter, startcol, defines the first colour number of the sequence of 7 to be used by the Font Manager for the anti-alias palette (7 + the background make up the total of 8 levels of shading). We have used colour 9 for this parameter in order to have a contiguous set of colour numbers for the palette.

When the program is run, the Font Manager will create the anti-alias palette by re-defining logical colours 8 - 15 as a set of shades intermediate between the so-called start and end colours. These 8 new shades will constitute the anti-aliasing palette. The start and end colours are defined in the remaining 6 parameters of the call, and will normally be the physical colours required by the user for the background and foreground of his text. These 6 parameters are made up of two red-green-blue sets, each of whose parameters range in value from 0 to 240 in steps of 16 (just as in the COLOUR n,r,g,b command - see RISC User Issue 2 page 24).

Line 180 uses the operating system call "Font_FindFont" to tell the Font Manager to set up and cache the Trinity font in point size 20 by 40. The last call, made to "Font_Paint", displays the text in the required font. The last two parameters of the

Arrays and Matrices

Mike Williams investigates the powerful matrix operations provided in ARM Basic.

Not only are arithmetic operations on the Archimedes very fast, but Basic V has been significantly enhanced to simplify the programming of matrix operations. As a result, many routines which would previously have required the use of nested FOR-NEXT loops and the like can now be programmed by simple assignments. For example, if we assume that a three dimensional array has been defined as:

```
DIM array(20,20,20)
```

and that we now wish to assign the value -1 to each element, then with no matrix operations we would have to write:

```
FOR i=1 TO 20
     FOR j=1 TO 20
          FOR k=1 TO 20
                 array(i,j,k)=-1
NEXT k,j,i
```

Instead, using ARM Basic we can write:

```
array()=-1
```

Not only is this much shorter, but an amazing 70 times faster!

In this short article I want to summarise some of the more useful features of Basic V for manipulating real or integer matrices (string arrays are allowed, but most of the following operations are just inappropriate). When an array is dimensioned, all its elements are set to zero, but any value n may be assigned to each element by writing:

```
array()=n
```

If n is to be replaced with any kind of expression, then this will need to be completely enclosed in parentheses, e.g.:

```
array()=(RND(1)*RND(1))
```

Otherwise some form of error message will result. If you need to increment or decrement all the elements of an array, then the '+=' or '–=' notation will work with arrays. For example:

```
array()+=SIN(x)
```

would increment all elements of the array by the current value of SIN(x).

All elements of an array may have the same quantity added, subtracted, multiplied or divided using the format:

```
array()=array() <operation> <expression>
```

where <operation> is any of +, –, * or / and <expression> is any valid arithmetic expression, enclosed in parentheses if it involves any operations. For example:

```
array()=array()/(2*PI)
```

would divide each element of array() by the value 2*PI.

You can also operate similarly on two arrays, but all arrays involved must have the same dimensions or an error will result. Such matrix operations take the form:

```
array1()=array2() <operation> array3()
```

where <operation> is again +, –, * or /. In this case, the operation applies to corresponding elements, so that:

```
array1()=array2()*array3()
```

would multiply corresponding pairs of elements together.

True matrix multiplication is also possible using the '.' symbol (dot product). In all cases, the numbers of elements in appropriate rows and columns must be the same - this derives from the principles of matrix multiplication which is beyond the scope of this article.

One other major improvement in Basic V is that arrays may now be legitimately included as the parameters of procedures and functions, and declared as LOCAL if required. Arrays so declared must also be dimensioned within the procedure or function definition, but this is not necessary for arrays passed as parameters.

A number of other facilities have also been provided. DIM may be used dynamically to determine either the number of dimensions of an array, or the size of any dimension. For example:

```
PRINT DIM(array())
```

would display the number of dimensions, while:

```
PRINT DIM(array(),1),DIM(array(),2)
```

would display the size of array() as the number of rows followed by the number of columns. This information can also be used to dimension one array to be the same size as another, e.g.:

```
DIM matrix(DIM(array(),1),DIM(array(),2))
```

which would dimension matrix to be the same size as array. Lastly, the new keyword SUM will sum all the elements of an array.

Basic V matrix operations clearly provide a powerful and convenient way of manipulating arrays. There are some omissions - you can't specify 'slices' (a particular row or column), which is a pity. Nevertheless, these new features will amply repay investigation and experiment. They are reasonably well described in the User Guide, but make sure you refer to the Addendum (if supplied).

RU

ARCHIMEDES VISUALS

This month's collection of visual effects includes routines to generate quadruple size shadowed text, and two different kinds of 3D backgrounds.

QUAD-SIZED SHADOWED TEXT

This routine is handy for producing smart looking titles and headings, without the need to resort to Acorn's Fancy Fonts (dealt with elsewhere in this issue). The anti-alias fonts have a considerable overhead in disc storage and loading time, and are also the subject of strict copyright control. The method used here is compact and easy to use. It works in all modes but 3, 6 and 7, though the selection of colours used in the accompanying program is not appropriate for 256 colour modes, and the 3D effect may only be obtained in graphics modes with four or more colours.

Whatever mode is in use (apart from 3, 6 or 7), the routine will double both the height and width of the normal text font. It is called with the format:

```
        PROCquad("Quad sized text")
```

or with:

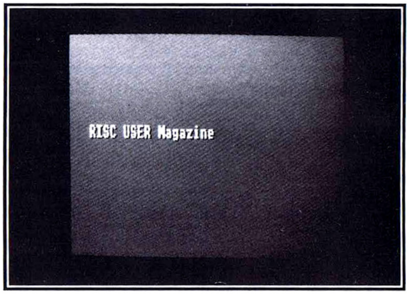
```
        PROCquad(text$)
```

where *text$* holds the text to be printed. Before the call is made, the text cursor should be moved to the required start of text position using, for example:

```
        PRINT TAB(X,Y);
```

If you are going to use the routine in your own programs, you will need lines 160 to 370. These contain a pair of procedures which read the computer's currently selected font (not Fancy Font), and re-define four new characters (characters 247-250) to make up a new extra large letter. This is repeated with each character output by the program. In addition you will need lines 50 and 60, which dimension two arrays, reserve some space, and define a string variable.

The program which accompanies the routine makes use of PORCquad in a special way. Here we have called the procedure twice with the same string of text, writing it in two different colours, and to two slightly different positions to give a shadowed, or 3D effect. In order to do this we have not used TAB to set the position of the text. Instead we have invoked VDU5 which causes text to be printed at the graphics cursor. Then, in lines 100 and 120, we have MOVEd the graphics cursor to the required positions, immediately prior to calling PROCquad.



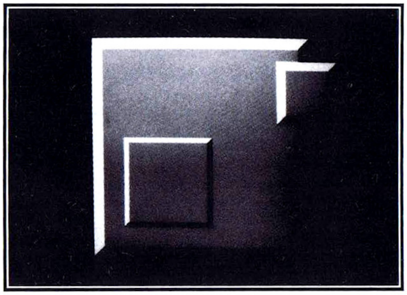
The shadowed effect is achieved by writing first in a dark colour and then in a brighter one. You will see from line 80 that the dark colour used (colour 8) is not black (0,0,0) but a dark red (80,0,0). This, against a salmon pink background (colour 9), ensures that the shadow will not be too harsh. Similarly, the foreground (colour 10) is not a pure white, but a slightly off-white (defined with RGB components 192,192,192 in line 90). The size of the shadow is given by the degree of offset between the two MOVE statements referred to earlier. The parameters currently used work well in the 40 and 80 column modes, but you may wish to increase the offset for mode 2. You will also see that in mode 2, the present text string is too long, and overflows the line. Incidentally the routine uses OSWORD 10 to read the Archimedes' character definitions, and this correctly copes with the various fonts held by the machine, and selected with *ALPHABET.

```
10 REM >Quad5
20 REM Quad sized shadowed text
30 REM By Lee Calcraft
40 :
50 DIM B(8),C(8),D%10
```

```
   60 quad$=CHR$(248)+CHR$(247)+CHR$(10)
+CHR$(8)+CHR$(8)+CHR$(250)+CHR$(249)+CHR
$(11)
   70 MODE12:VDU5
   80 COLOUR9,208,80,80:COLOUR8,80,0,0
   90 COLOUR10,192,192,192:GCOL128+9:CLG
  100 GCOL8:MOVE 100,600
  110 PROCquad("RISC USER Magazine")
  120 GCOL10:MOVE94,604
  130 PROCquad("RISC USER Magazine")
  140 VDU4:END
  150 :
  160 DEFPROCquad(text$)
  170 LOCAL A,I,N,Q,R,S
  180 FOR A=1 TO LEN(text$)
  190 PROCchread(ASC(MID$(text$,A,1)))
  200 FOR Q=0 TO 1:FOR N=0 TO 7
  210 C(N)=0
  220 FOR I=0 TO 3
  230 C(N)=C(N)-(2^(2*I)+2^(2*I+1))*((B(
N) AND 2^(I+4*Q))>0)
  240 NEXT:NEXT
  250 FOR R=0 TO 1
  260 S=4*R
  270 VDU23,247+2*R+Q,C(S),C(S),C(S+1),C
(S+1),C(S+2),C(S+2),C(S+3),C(S+3)
  280 NEXT:NEXT
  290 PRINTquad$;
  300 NEXT:ENDPROC
  310 :
  320 DEF PROCchread(I)
  330 LOCAL A:?D%=I
  340 SYS 7,&A,D%
  350 FOR A=0 TO 7
  360 B(A)=D%?(A+1)
  370 NEXT:ENDPROC
```

THE PLINTH EFFECT

The second program generates the square "plinth" shape shown in the accompanying photograph. As you will see from the listing, the routine for generating the effect has been parcelled up into a procedure for ease of use. PROCplinth has a total of 7 parameters. The first two are the graphics coordinates of the bottom left corner of the plinth. The next is its width, and the fourth the number of graphics units between the inner and the outer boxes. Finally come the three numbers of the colours used for the plinth. They are supplied in the order: top left edges, front face, bottom right edges. In our program, these colours are set up in lines 60 to 80. To achieve the desired effect, some considerable care is needed in the choice of colour.



```
   10 REM >Plinth3
   20 REM Raised Plinth
   30 REM By Lee Calcraft
   40 :
   50 MODE12
   60 COLOUR9,192,192,192
   70 COLOUR10,144,144,144
   80 COLOUR11,80,80,80
   90 :
  100 PROCplinth(0,0,1000,1000,50,9,10,1
1)
  110 PROCplinth(850,612,280,280,40,9,10
,11)
  120 PROCplinth(150,150,400,400,20,9,10
,11)
  130 END
  140 :
  150 DEFPROCplinth(X,Y,WX,WY,W,C1,C2,C3
)
  160 GCOLC1:RECTANGLE FILL X,Y,WX,WY
  170 GCOLC3:MOVE X,Y:MOVE X+WX,Y
  180 PLOT85,X+WX,Y+WY
  190 GCOL7:LINE X,Y+WY,X+W,Y+WY-W
  200 GCOLC2
  210 RECTANGLE FILL X+W,Y+W,WX-2*W,WY-2
*W
  220 ENDPROC
```

Continued on page 30

Next month's RISC User will feature a "Colour Editor" allowing the easy selection of colours for effects such as this.

RISC USER MEMORY EDITOR

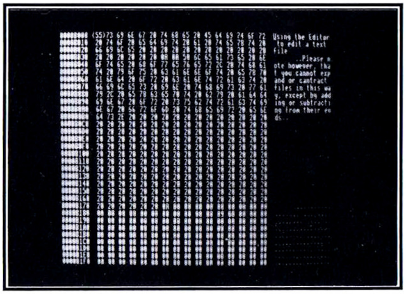
By Barry Christie

O.S. 0.2, 0.3, 1.2

Display and edit the contents of user RAM with the RISC User Memory Editor.

The accompanying program, though relatively short, provides a full screen display of user memory in both hex and ASCII format, with scrolling in both directions, and with editing facilities in both hex and ASCII. In these respects it offers considerable advantages over the features provided by the debugging module, which is assembler orientated, and which does not provide full screen editing and bi-directional scrolling.

To make use of the Memory Editor, first type in the program and save it away. When it is run, it requests a starting memory address in hex within a given range. The range offered is from zero to just below the machine's current setting of HIMEM. If Return is pressed on a null entry, the start address will be set to the current value of PAGE.



```
[illegible]000 (55)73 69 6E 67 20 74 68 65 20 45 64 69 74 6F 72  Using the Editor
[illegible]010  20 74 6F 20 65 64 69 74 20 61 20 74 65 78 74 20   to edit a text
[illegible]020  66 69 6C 65 20 20 20 20 20 20 20 20 20 20 20 20  file
[illegible]030  20 20 20 20 20 20 0A 0A 50 6C 65 61 73 65 20 6E        ..Please n
[illegible]040  6F 74 65 20 68 6F 77 65 76 65 72 2C 20 74 68 61  ote however, tha
[illegible]050  74 20 79 6F 75 20 63 61 6E 6E 6F 74 20 65 78 70  t you cannot exp
[illegible]060  61 6E 64 20 6F 72 20 63 61 6E 74 72 61 63 74 20  and or cantract
[illegible]070  66 69 6C 65 73 20 69 6E 20 74 68 69 73 20 77 61  files in this wa
[illegible]080  79 2C 20 65 78 63 65 70 74 20 62 79 20 61 64 64  y, except by add
[illegible]090  69 6E 67 20 6F 72 20 73 75 62 74 72 61 63 74 69  ing or subtracti
[illegible]0A0  6E 67 20 66 72 6F 6D 20 74 68 65 69 72 20 65 6E  ng from their en
[illegible]0B0  64 73 2E 20 20 20 20 20 20 20 20 20 20 20 20 20  ds.
[illegible]0C0  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]0D0  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]0E0  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]0F0  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]100  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]110  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]120  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]130  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]140  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]150  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]160  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[illegible]170  20 20 20 20 20 20 0A [illegible]
[illegible]
```

As you can see from the accompanying illustration, the display has three fields. These hold RAM addresses in 16 byte steps, and RAM contents in hex and ASCII respectively. In the case of the latter, unprintable ASCII codes are represented by a dot. To move around in RAM, use the cursor keys alone, or together with the Ctrl key. There are just two further keys to remember: Escape will terminate the program, while the "Print" key (to the right of function key f12) will send the currently displayed screen to your printer.

When the Editor is run it starts off in hex editing mode, signified by the parentheses "()" around the active RAM location. To edit RAM, just enter new values in hex from the keyboard, high byte first. The digit entered will appear first in the low nibble position, and then be shifted to the left if a second entry is made. To edit in ASCII, press Tab, and you will see the parentheses altered to square brackets. Now, locations may be altered one byte at a time using any keyboard character. You should take great care about altering RAM, however, because if you alter the RAM which is used by the program itself, the program will crash. If this happens you should use Ctrl-Break to clear your machine, and then reload the program.

The Editor may easily be used to examine or edit the contents of any disc file. The best way to do this is to exit from the Editor by pressing Escape, then load the target file into RAM using:

```
*LOAD filename B000
```

The location &B000 is well clear of the Editor program (providing that you have not altered PAGE from its default). Then run the Editor, and give B000 as the RAM address to be examined. If you wish to resave an altered file, leave the Editor by pressing Escape, and type:

```
*SAVE filename B000 nnnn
```

where nnnn is the address in memory of the end of your file.

```
 10 REM              >MemEdit4
 20 REM Program      Memory Editor
 30 REM Version      A 0.4
 40 REM Author       Barry Christie
 50 REM RISC User Jan/Feb 1988
 60 REM Program      Subject to copyright
 70 :
 80 MODE0:VDU19,0,24,174,174,174
 90 ON ERROR PROCerror:END
100 PROCinitialise
110 PROCmemoryeditor
120 END
130 :
  140 DEF PROCmemoryeditor
  150 REPEAT
  160 *FX 21,0
  170 key%=GET
  180 CASE key% OF
  190 WHEN   0  : PROCcheckprint
  200 WHEN   9  : PROCtabkey
  210 WHEN 136  : PROClineLL
  220 WHEN 137  : PROClineRR
  230 WHEN 138  : PROClineDD
  240 WHEN 139  : PROClineUU
  250 OTHERWISE : PROCbytechange
  260 ENDCASE
  270 UNTIL FALSE
  280 ENDPROC
  290 :
  300 DEF PROClineLL
  310 IF INKEY(-2) THEN PROCpageLL ELSE
PROClinesub(1)
  320 ENDPROC
  330 :
  340 DEF PROClineRR
  350 IF INKEY(-2) THEN PROCpageRR ELSE
PROClineadd(1)
  360 ENDPROC
  370 :
  380 DEF PROClineUU
  390 IF INKEY(-2) THEN PROCpageUU ELSE
PROClinesub(16)
  400 ENDPROC
  410 :
  420 DEF PROClineDD
  430 IF INKEY(-2) THEN PROCpageDD ELSE
PROClineadd(16)
  440 ENDPROC
  450 :
  460 DEF PROCpageLL
  470 PROCcursor(1):curpos%-=curpos% MOD
 16:PROCcursor(tabkey%)
  480 ENDPROC
  490 :
  500 DEF PROCpageRR
  510 PROCcursor(1):curpos%+=16-curpos%
MOD 16-1:PROCcursor(tabkey%)
  520 ENDPROC
  530 :
  540 DEF PROCpageUU
  550 memory%-=512
  560 IF memory%>=bound1% THEN PROCpaged
isplay(FALSE) ELSE memory%+=512:VDU7
  570 ENDPROC
  580 :
  590 DEF PROCpageDD
  600 memory%+=512
  610 IF memory%<=bound2% THEN PROCpaged
isplay(FALSE) ELSE memory%-=512:VDU7
  620 ENDPROC
  630 :
  640 DEF PROClinesub(changes%)
  650 PROCcursor(1)
  660 curpos%-=changes%
  670 IF curpos%<=curmin% THEN
  680 memory%-=16
  690 IF memory%<bound1% THEN
  700 curpos%+=changes%:memory%+=16:VDU7
  710 ELSE
  720 VDU30,11:curpos%+=16
  730 PROClinedisplay(0,memory%)
  740 ENDIF
  750 ENDIF
  760 PROCcursor(tabkey%)
  770 ENDPROC
  780 :
  790 DEF PROClineadd(changes%)
  800 PROCcursor(1)
  810 curpos%+=changes%
  820 IF curpos%>=curmax% THEN
  830 memory%+=16
  840 IF memory%>bound2% THEN
  850 curpos%-=changes%:memory%-=16:VDU7
  860 ELSE
  870 VDU31,0,31,10:curpos%-=16
  880 PROClinedisplay(31,memory%+496)
  890 ENDIF
  900 ENDIF
  910 PROCcursor(tabkey%)
  920 ENDPROC
  930 :
  940 DEF PROClinedisplay(taby%,address%
)
  950 part1$=" ":part2$="  ":OFF
  960 FOR bytes%=0 TO 15
  970 byte%=bytes%?address%
  980 part2$+=FNnumtochr(byte%):part1$+=
" "+FNnumtostr(byte%,2)
  990 NEXT bytes%
 1000 PRINT TAB(3,taby%)FNnumtostr(addre
ss%,7);part1$;part2$;:ON
 1010 ENDPROC
 1020 :
```

→

```
 1030 DEF PROCpagedisplay(printer%)
 1040 IF printer% THEN VDU2
 1050 FOR line%=0 TO 31
 1060 PROClinedisplay(line%,memory%+line
%*16):IF printer% THEN VDU1,10
 1070 NEXT line%
 1080 VDU3:PROCcursor(tabkey%)
 1090 ENDPROC
 1100 :
 1110 DEF PROCcheckprint
 1120 printkey=GET
 1130 IF printkey=128 THEN
 1140 COLOUR 0:COLOUR 129:PRINT TAB(3,31
)"PRINTING";
 1150 COLOUR 1:COLOUR 128:PROCpagedispla
y(TRUE)
 1160 ENDIF
 1170 ENDPROC
 1180 :
 1190 DEF PROCbytechange
 1200 PROCtabxy:IF tabkey% THEN PROCkeyv
alue ELSE PROChexvalue
 1210 ENDPROC
 1220 :
 1230 DEF PROCtabxy
 1240 char%=curpos% MOD 16:tabx%=char%*3
+11:taby%=curpos% DIV 16
 1250 ENDPROC
 1260 :
 1270 DEF PROCkeyvalue
 1280 PROCprintvalue:PROClineRR
 1290 ENDPROC
 1300 :
 1310 DEF PROChexvalue
 1320 key%=INSTR("00112233445566778899Aa
BbCcDdEeFf",CHR$key%)-1
 1330 IF key%>=0 THEN key%=key%>>>1:key%
+=(curpos%?memory% AND &0F)<<4:PROCprint
value
 1340 ENDPROC
 1350 :
 1360 DEF PROCprintvalue
 1370 curpos%?memory%=key%
 1380 PRINT TAB(tabx%+1,taby%)FNnumtostr
(key%,2);
 1390 PRINT TAB(chrpos%+char%,taby%)FNnu
mtochr(key%);
 1400 ENDPROC
 1410 :
 1420 DEF PROCtabkey
 1430 tabkey%=NOT(tabkey%):PROCcursor(ta
bkey%)
 1440 ENDPROC
 1450 :
 1460 DEF PROCcursor(bracket%)
 1470 CASE bracket% OF
 1480 WHEN -1 : bracket$="("+skip$+")"
 1490 WHEN  0 : bracket$="["+skip$+"]"
 1500 WHEN  1 : bracket$=" "+skip$+" "
 1510 ENDCASE
 1520 PROCtabxy:PRINT TAB(tabx%,taby%)br
acket$;
 1530 VDU31,chrpos%+char%,taby%
 1540 ENDPROC
 1550 :
 1560 DEF FNnumtochr(number%)
 1570 IF number%<32 OR number%>126 THEN
="." ELSE =CHR$number%
 1580 :
 1590 DEF FNnumtostr(number%,length%)
 1600 =STRING$(length%-LEN(STR$~(number%
)),"0")+STR$~(number%)
 1610 :
 1620 DEF PROCinitialise
 1630 PRINT''"Please enter required star
t address"
 1640 PRINT"or press Return to start at
PAGE"
 1650 PRINT"Acceptable range 0 to ";~(HI
MEM-&200)SPC8;
 1660 INPUT"&"memory$:CLS:ON
 1670 IF memory$="" THEN memory%=PAGE EL
SE memory%=EVAL("&"+memory$)
 1680 COLOUR 0,120,0,32
 1690 OSCLI("FX 4,1"):OSCLI("FX 225,2")
 1700 tabkey%=-1:skip$=CHR$9+CHR$9
 1710 curpos%=0:curmin%=-1:bound1%=0
 1720 chrpos%=61:curmax%=512:bound2%=HIM
EM-512
 1730 PROCpagedisplay(FALSE)
 1740 ENDPROC
 1750 :
 1760 DEF PROCerror
 1770 VDU7,31,3,31:PRINT:COLOUR 0:COLOUR
 129
 1780 OSCLI("FX 4,0"):OSCLI("FX 225,1")
 1790 IF ERR=17 THEN PRINT "FINISHED ";
ELSE REPORT:PRINT " at line ";ERL;" ";
 1800 COLOUR 0,0,0,0:COLOUR 7:COLOUR 128
 1810 ENDPROC
```

RU

ARCHIMEDES DISC MENU (Part 3)

By Lee Calcraft

Notes on using the special auto-load option on David Pilling's Disc Menu.

As we mentioned in the first part of this series, the RISC User Menu has a special facility engaged with the middle button of the mouse, which allows you to select files from the menu, and automatically engage a word processor, or other software.

What happens when you click the middle button on a filename is that the Menu stores the selected filename in the operating system variable FNAME. It then tries to EXEC in a file called !MENU*. This means that the user can create one or more !MENU files on his disc which load the required application software, such as a word processor, or whatever, then set up function keys, assign the date, set up printer options, etc. And finally, the EXEC file can pick up the filename selected by the user (and stored by the Menu in the variable FMENU), and cause this to be loaded into the application.

```
*| >!MENUedit
*| Auto Loader for ARMBE
*| Version 0.3
OSCLI("KEY1 LOAD "+CHR$34+"<FNAME
  >"+CHR$34+"|MEDIT|M")
*FX138,0,129
```

The system is completely flexible, and can be used with almost any application. To illustrate its use, a number of examples are given. The first causes the selected file to be loaded into the Basic Editor. As with each of the examples, it should be saved as an EXEC file, that is to say, without line numbers (either use *BUILD or a word processor or text editor), and should be given a file type of &FFE (type *SETTYPE filename FFE). The name under which it is saved must begin with !MENU, and could be followed by the letters edit for example, as a reminder of its function. The file !MENUedit should be saved in the same directory as the one containing the files which it will be used to load; and note that no directory should contain more than one !MENU file.

```
*| >!MENUww+
*| Auto Loader for Wordwise Plus
*| Version 0.3
*BUFFER ON
OSCLI("KEY1 :D$="+CHR$34+"<SYS$D
  ATE> <SYS$YEAR>"+CHR$34+"|M:F$
  ="+CHR$34+"<FNAME>"+CHR$34+"|M
  :LOAD TEXT F$|M|M")
*$.MODULES.65ARTHUR
*$.MODULES.WW+
*FX138,0,129
```

The second example, which might be saved as !MENUww+, loads files into Wordwise Plus, and gives some idea of the flexibility of the technique. It first turns on a printer buffer, and causes the Emulator to be loaded from disc, then Wordwise itself. Next it copies the system date into Wordwise's variable D$, and the user-selected filename into the Wordwise filename variable F$, and then loads the file. If you wish to set up your printer or function keys, this can be performed at the very start of the EXEC file. The third example loads any program into the Twin text editor, with the LISTO option 8 set, and with white text on a blue background.

```
*| >!MENUtwin
*| Auto Loader for Twin
*| Version 0.3
*SET ALIAS$Blue ECHO |<19>|<0>|<4>
  |<0>|<0>|<0>
OSCLI("KEY3 LOAD "+CHR$34+"<FNAME>
  "+CHR$34+"|MTWINO8|M"+CHR$129+"B
  lue|M|M")
*FX138,0,131
```

The whole system is quite flexible, so that you may customise it to meet individual needs. If you create any interesting !MENU files, please let us know so that we may pass them on to other readers. RU

OVER THE RAINBOW

by Mike Williams

We continue our series on the graphics capabilities of the Archimedes by delving into the complex secrets of the 256 colour modes.

Last month we looked at how we can control the choice of colour in those Archimedes modes which allow up 16 colours on the screen at any one time. This time we will look at how things are managed in the 256 colour modes, that is modes 10, 13 and 15. This is quite different to what happens in other modes. Indeed, it would appear on first examination that only 64 colours are possible, not the full 256.

The problem is that on the older BBC micros, logical colours in the range 0 to 127 (of which only 0 to 15 were used), specify a *foreground* colour, and logical colours 128 to 255 specify a *background* colour. It is Acorn's decision to maintain compatibility with that system which is at the root of our immediate problems. If we were to specify a colour above 127, then the convention already established would deem that this was a background colour.

USING 64 COLOURS

In practice, COLOUR or GCOL are used in the 256 colour modes, just as in all the other modes, to specify a logical colour number, but in the range 0 to 63 (foreground). Adding 128 to any of these specifies a background colour just as before. This gives us, without any further effort, a range of 64 colours. For example:

```
        COLOUR 15
```

would select a shade of yellow for text, while:

```
        GCOL 148
```

would select a deep blue-green (colour 20 - since 148 = 128 + 20) as a graphics background colour.

USING 256 COLOURS

In order to increase the number of colours from 64 up to the maximum of 256 we need to use the additional keyword TINT. The value of TINT may be used to add one of four levels of white to each of the basic 64 colours. The standard format for selecting one of the 256 colours is then:

```
        COLOUR n TINT t
```

or:

```
        GCOL n TINT t
```

where n is the logical colour number in the range 0 to 63 (128 to 191 for background colours), and t is the tint level.

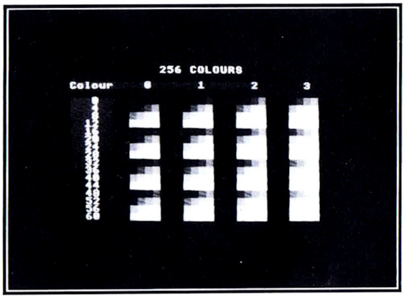
Now although there are only four levels of tint, the value of TINT can be anywhere in the range 0 to 255 (not 0 to 3 as given in early User Guides) with the following meaning:

range	meaning
0-63	level 0 (min.)
64-127	level 1
128-192	level 2
192-255	level 3 (max.)

In practice, it is therefore best to use the values 0, 64, 128 and 192 when determining the level of tint. The current (or default) level of tint is *always* applied to all colours specified even if TINT is left out.

This default is set at 0 (minimum white) initially. In such cases all colours, including white, appear slightly dull on the screen. To get the brightest white (colour 63) you will need to set the tint level to 192, though you will then find that black (colour 0) appears as very dark (but visible) grey.

There are a few points to note. To ensure the correct shade of colour I suggest that you always use TINT with GCOL or COLOUR, unless you are happy with the just the basic 64 colours. Note too, that tint levels exist independently for text and graphics. Thus specifying TINT in a COLOUR command will not affect the level of tint applied in a subsequent GCOL command, and vice versa. lastly, the commands *POINTER and MOUSE ON appear to have the side effect of setting the tint level to its maximum (192).



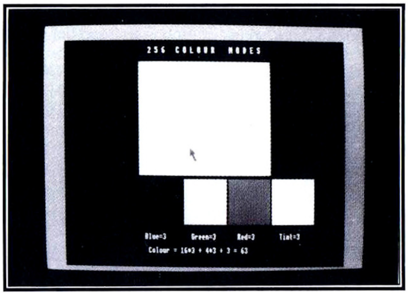
The accompanying program, Shades256, displays (in mode 13) all 256 colours on the screen together, and you may find this a useful colour reference chart for modes 10, 13 and 15. The colours are displayed in tabular form, and in blocks of four tints at a time. The colour (i.e. a number in the range 0 to 63) for any block is given by adding together the appropriate row and column numbers. In each block of four, each shade has increasing amounts of tint added from left (darkest) to right (brightest).

Essentially, the display is produced by two nested loops, the outer one cycling through all 64 basic colour numbers, and the inner one cycling through the four levels of tint (lines 130 and 140). Each colour is displayed by selecting it as a background colour (using COLOUR and TINT), and then printing a space. The remaining statements are concerned with the general layout and design of the screen display. To exit from the program just press Escape.

```
   10 REM >Shades256
   20 REM Program Shades256
   30 REM Version A1.0
   40 REM Author  Mike Williams
   50 REM RISC User Jan/Feb 1988
   60 REM Program subject to copyright
   70 :
   80 MODE 13:OFF:ON ERROR GOTO 250
   90 PRINTTAB(14,1)"256 COLOURS"
  100 PRINT'TAB(2);:COLOUR 148 TINT 64:C
OLOUR 15 TINT 64
  110 PRINT"Colour"TAB(12)"0";TAB(19)"1"
;TAB(26);"2"TAB(33);"3";SPC3
  120 COLOUR 128 TINT 0
  130 FOR shade=0 TO 63
  140 FOR tint=0 TO 192 STEP 64
  150 IF shade MOD4=0 AND tint MOD256=0
THEN PRINT'TAB(2);:COLOUR 148 TINT 64:PR
INTSPC(2-(shade<9));STR$((shade DIV4)*4)
;SPC2;:COLOUR 128 TINT 0:PRINTTAB(10);
  160 COLOUR 128+shade TINT tint
  170 PRINT" ";
  180 NEXT tint
  190 COLOUR 128 TINT 0:PRINT SPC3;
  200 NEXT shade
  210 PRINT''TAB(2);:COLOUR 148 TINT 64:
PRINT STRING$(35,CHR$32)
  220 REPEAT UNTIL FALSE
  230 END
  240 :
  250 MODE12:REPORT:PRINT" at line ";ERL
:END
```

256 COLOUR MODES EXPLAINED

It is quite instructive to examine in more detail just how the colours are determined in a 256 colour mode. The following program, Mixer256, will help in this task. Superficially, it resembles last month's Colour Mixing program, but it cannot be emphasized too much that the 256 colour modes operate quite differently to the two, four and sixteen colour modes.



Each of the 64 basic colours is made up of red, green and blue components. Each of these may be set at one of four levels (in the range 0 to 3). The colour number may then be computed in the following way:

```
colour = 16*blue + 4*green + red
```

In other words, the use of COLOUR or GCOL is linked to a byte in memory (8 bits). The bottom two bits specify the level of red, the next two the level of green and the next two again the level of blue. The top two bits do not specify any colour at all, but the top-most bit signifies foreground (if 0) or background (if 1).

In addition, TINT is linked to a further memory byte, and it is the top two bits of this which determine its level. In this context it is more convenient to think of the tint as being in the range 0 to 3, as with the colours, and then multiplying the value in this range by 64 to specify TINT correctly. Thus COLOUR (or GCOL) would take the form:

```
COLOUR 16*blue+4*green+red TINT 64*tint
```

When running the program, the large square shows the currently selected shade of colour (in the range 0 to 63), and the three squares below indicate the levels of red, green and blue. To the right of these three, a further square shows the level of tint. Initially, the program selects colour 63 (white) and sets the tint to 3 (maximum whiteness) to give a bright white result. The mouse pointer may be moved to any of the squares on the screen, and pressing the left-most *(select)* button will decrease the colour value of that square by one, and pressing the right-hand button *(adjust)* will increase that colour value by one.

Clicking on the large square will not only change that colour shade (up or down), but will adjust the levels of red, green and blue accordingly. Alternatively, clicking on the individual colour squares will directly increase or decrease the levels of these three colour components (and the changes will be reflected in the colour of the large square), while clicking on the tint square will independently change this. The levels of all colours and tint are continuously updated on the screen, and displayed in numerical form as well.

As you will have seen, there is much more to using the 256 colour modes on the Archimedes than might at first be expected. Whether your own applications of these modes demands simple or complex colour handling you should find that the two programs presented here will help.

In the above examples, we have not at any time re-defined any of the physical to logical colour assignments (as we did last month in other modes with VDU19). Each shade displayed has been obtained simply by specifying the appropriate colour number and tint level required. It is possible to re-assign the physical colours used in these modes, but it is a complex process, and one which Acorn advises against whenever possible.

```
   10 REM >Mixer256
   20 REM Program   Mixer256
   30 REM Version   A1.7
   40 REM Author    Mike Williams
   50 REM RISC User Jan/Feb 1988
   60 REM Program subject to copyright
   70 :
   80 ON ERROR GOTO 670
   90 MODE15:OFF:*POINTER
  100 VDU19,1,24,240,96,0
  110 VDU19,1,25,112,96,112
  120 VDU19,2,25,240,0,112
  130 COLOUR 15 TINT 192
  135 COLOUR 148 TINT 64
  140 PRINTTAB(22,1)"  2 5 6   C O L O U
R   M O D E S  "
  150 red=3:green=3:blue=3
  155 shade=63:tint=3
  160 REPEAT
  170 PROCmouse
  180 PROCboxes
  190 TIME=0:REPEAT UNTIL TIME>5
  200 UNTIL FALSE
  210 END
  220 :
  230 DEFPROCboxes
  240 GCOL shade TINT 64*tint
  250 RECTANGLEFILL 345,420,590,500
  260 FOR box=1 TO 4
  270 PROCrect(box)
  280 NEXT box
  290 ENDPROC
  300 :
  310 DEF PROCrect(box)
  320 GCOL FNrgb(box) TINT 64*tint
  330 RECTANGLEFILL 200*box+145,200,190,
200
  340 PRINTTAB(23,27)"Blue=";blue;TAB(36
,27)"Green=";green;TAB(49,27)"Red=";red;
TAB(61,27)"Tint=";tint
  350 PRINTTAB(24,29)"Colour = 16*";blue
;" + 4*";green;" + ";red;" = ";FNjust(sh
ade,2)
  360 ENDPROC
  370 :
  380 DEF PROCmouse
  390 LOCAL p,q,x,y,state
  400 MOUSE x,y,state:p=(5-2*state)/3
  410 CASE state OF
  420 WHEN 1,4
  430 IF FNTest(x,y,345,420,590,500) THE
N shade=(shade+p+64)MOD64:PROCsetrgb
  440 IF FNTest(x,y,345,200,190,200) THE
N blue=(blue+p+4)MOD4
  450 IF FNTest(x,y,545,200,190,200) THE
N green=(green+p+4)MOD4
  460 IF FNTest(x,y,745,200,190,200) THE
N red=(red+p+4)MOD4
  470 IF FNTest(x,y,945,200,190,200) THE
N tint=(tint+p+4)MOD4
  480 ENDCASE
  490 shade=16*blue+4*green+red
  500 ENDPROC
  510 :
  520 DEF FNrgb(box)
  530 IF box=4 THEN =63
  540 =shade AND (3*2^(6-2*box))
  550 :
  560 DEF PROCsetrgb
  570 red=FNrgb(3)
  580 green=FNrgb(2)/4
  590 blue=FNrgb(1)/16
  600 ENDPROC
  610 :
  620 DEF FNjust(p,w)=STR$(p)+STRING$(w-
LEN(STR$(p)),CHR$32)
  630 :
  640 DEF FNTest(x,y,x1,y1,w,h)
  650 IF x>=x1 AND x<=x1+w AND y>=y1 AND
 y<=y1+h THEN =TRUE ELSE =FALSE
  660 :
  670 MODE12:REPORT:PRINT" at line ";ERL
:END
```

RU

REVIEWS

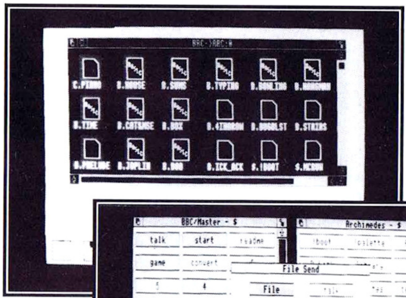
BEEB TO ARCHIMEDES LINKS

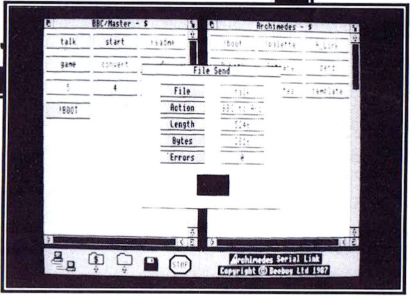
Reviewed by Dennis Weaver

Many Archimedes users have upgraded from the BBC Micro or Master, and as a consequence, have quantities of software on 5.25 inch discs which they would like to transfer to the Archimedes. To meet this need a number of suppliers have announced RS232 transfer links, while others have proposed connecting 5.25 inch drives to the Archimedes. In this brief review, we will be looking at two packages which follow the former route.

BRAINSOFT'S 'DISC CONVERSION KIT'

In spite of their somewhat unlikely name, Brainsoft have produced a competent, and keenly priced package. For just £14 (no VAT), you get a transfer lead and two discs, one for the Beeb, and one for the Archimedes. The manual is brief and reasonably clear, and the lead, though only 1 metre in length, is well made.





The principle used by the Brainsoft package is to allow the BBC Micro with its disc drive (DFS format only at the moment) to behave as an additional disc drive on the Archimedes. When you boot the discs as instructed, the Archimedes displays a modified version of Acorn's Desktop (copyright Acorn!). If you click on its extra disc icon, your BBC disc is catalogued, and you may then use the mouse to mark any number of files for transfer. On clicking the *menu* button, a series of options is offered, including one to transfer all marked files to the Archimedes 3.5 inch disc.

All appears to function as described in the manual, except that I could not get the package to work with the new series one operating system. Once Brainsoft have upgraded their package to work with ADFS discs, and to cope with the series one operating system, they will have a most competitive product.

BEEBUG'S 'ARCHIMEDES SERIAL LINK'

At £17.25 this costs a little more than the Brainsoft unit, and contains just one 3.5 inch disc with lead and manual. But the software is more sophisticated, and the accompanying lead is somewhat longer at 3 metres in length. To get the BEEBUG unit running for the first time, you must boot the 3.5 inch disc on the Archimedes, and then type two FX commands on the BBC micro. The Beeb's part of the transfer software is then ported across from the Archimedes, and can be saved for subsequent use.

The BEEBUG package, which will handle DFS or ADFS format Beeb discs, provides the user with a dual window system working under WIMPs which allows you to move around the directories of either Beeb or Archimedes disc, and to mark any or all of the files (and/or directories) from either machine, using the mouse. When marking is complete, the files are automatically transferred to the second machine. Complete directories may be transferred in this way, and the software will automatically create new directories on the destination disc where necessary. Like the Brainsoft unit, the BEEBUG package permits the transfer of all types of file, and when transferring from Beeb to Archimedes, it also attempts to assign file types intelligently.

Disc Conversion Kit, £14.00
Brainsoft are on 01-486 0321

Archimedes Serial Link, £17.25 (Members)
BEEBUG are on (0727) 40303

RU

REVIEWS

SYSTEM DELTA PLUS

Mike Williams and Mark Sealey have compiled this report on Minerva's latest database for the Archimedes, System Delta Plus.

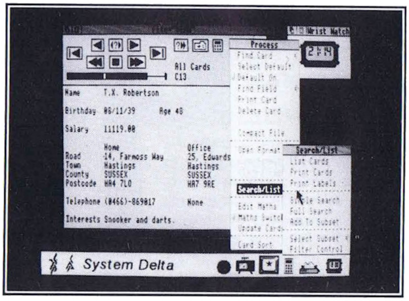
RISC User has already reviewed Minerva's earlier 'Deltabase' package (see the November issue), and a thoroughly workmanlike product it was found to be. Big brother, in the form of System Delta Plus, is now available but at a much higher price. This time Minerva has fully exploited the WIMP environment of the Archimedes to produce a highly attractive looking package.

Minerva claims that despite the relatively low price, System Delta Plus offers facilities as good as (if not better than) the likes of dBase, Lotus 123 and other PC software, plus the bonus of the enhanced graphics of the Archimedes. Those accustomed to the lower price of older BBC micro products may take a little more convincing.

Strictly speaking, System Delta Plus is a set of sophisticated database commands (star commands) which can be used in conjunction with Basic to build a tailor-made database. A complete system designed in this way and referred to as WIMP Card Index is supplied as part of System Delta Plus. As it is assumed that most users will buy the package for the WIMP Card Index, the accompanying manual, and this review, concentrate on this. For the more advanced user, the full power of System Delta Plus is always available for those applications that warrant it.

The manual - whilst being better than the one for Deltabase - is still a bit of a compromise. It consists of a snazzy plastic clip-binder containing over 120 pages of very ordinary looking text with a good few black and white illustrations. It is rather disappointing after the glossy and colourful cover. Nor is the organisation of the material particularly well thought out, although you'll find everything you need - eventually. There is an introductory tutorial, called the 'Experimental Section', and a rather short reference section at the back where all the System Delta Plus functions are listed with a brief explanation. On the whole we feel that the manual performs better as a reference when you are already familiar with the package and know where to find what you want.

This aside, the WIMP Card Index can be thoroughly recommended. Its capacity is impressive, though neither a standard 800K floppy nor even a 20MByte hard disc will provide anywhere near the storage needed for the maximum 2.14 billion records which the system is theoretically capable of handling. And fields are still limited to the maximum 255 character length permitted by Basic. Several files can be open at the same time, and it is quite possible to browse through one file while printing a report from another and searching a third. The system does slow down in these circumstances but this multi-tasking environment can be of real value when comparing data.



```
Name      T.X. Robertson
Birthday  06/11/39      Age 48
Salary    11119.00
          Home                Office
Road      14, Farmoss Way     25, Edwards
Town      Hastings            Hastings
County    SUSSEX              SUSSEX
Postcode  HA4 7LO             HA7 9RE
Telephone (0466)-869017       None
Interests Snooker and darts.
```

All of this is managed through a system of WIMP menus. These control all the usual database functions, including the option to divide any file into as many as 32 subsets, each of which may be treated quite independently of the others, and of the main file. At this level you can also perform mathematical and statistical operations on data.

A menu bar across the foot of the screen gives you constant access to a number of additional functions. These allow for the input of both Operating System and System Delta star commands as well as monitor and printer adjustments. A wrist-watch and pocket calculator of the usual simple style are also provided, and can be selected and positioned on the screen. These hardly seem essential adjuncts of any database system, and one is faced with the conclusion that Minerva has succumbed to the temptation to exploit the Archimedes graphics capability with some pretty but unnecessary images. Although the menu screens are remarkably uncluttered and easy to read, do you really want to be able to display and set a wristwatch?

In practice, the menu system works well. Clicking the menu button of the mouse anywhere on the empty screen produces a menu from which you can create a new file or open an existing one. Other global options are also available from this menu which allow you to branch sideways to other menus for sorting, data transfer and the creation or modification both of file structures themselves and layout formats.

Once a file has been opened, a window appears on the screen displaying the first record with a panel of control icons for browsing backwards and forwards through the records. Pressing the menu button with the pointer anywhere within this 'data window' produces a different menu with options appropriate to record handling (Find Field, Find Card, Delete Card etc), and several options can be followed to further sub-menus.

These menus are very easy to use, with prompts for confirmation when it matters. Naturally, when creating a new file, you can arrange for data fields to be positioned wherever you want in the data window, with separate text prompts of your choice. It is also possible to alter your record design later, including even the addition of extra fields if you wish. What is more, data held in a particular subset could also form the basis of a new file altogether. Nothing has to be re-entered.

The basic routines are all in ARM machine code. The full System Delta Plus is really a suite of overlays in Basic to tie them together. So the user with the right knowledge could customise and manipulate their own database by means of some 41 primary commands. For example:

```
*SDfind <string$> <file>
```

finds the card containing <string$>. By the time you read this, an Advanced Programmer's Manual for this product will also be available to help, although the majority of users will probably not want to delve this deep.

There is also an Import facility that handles data from Viewstore, the Computer Concepts Inter-series, BEEBUG's Masterfile, pure ASCII, Mini-Office, Alpha-Base, Beta-Base, dBase, Lotus, DataGem and the new BBC Uniform formats. Files can also be created that are downward (BBC System Delta) compatible.

CONCLUSION

The speed of System Delta Plus is impressive, and the constant but unobtrusive on-screen progress reports (when sorting for instance) add to the overall friendliness. For the experienced business user who switches, say, from dBase III to an Archimedes system, the WIMP Card Index will win hands down. It is easier to understand and control what's happening and certainly much quicker. One of the databases that we have tried is 175K long, has four fields and 600 records. It takes just four minutes to sort on three fields.

Data handling can be a very involved process, and Minerva have certainly gone for a very comprehensive package with System Delta Plus. Nevertheless, the package succeeds in being easy to understand, fast in operation, and enjoyable to use. System Delta Plus sets a high standard for database software on the Archimedes, and one that deserves to last for some time to come.

Product	System Delta Plus
Supplier	Minerva Systems 69 Sidwell Street, Exeter EX4 6PH.
Price	£69.95 inc. VAT

RU

A MEMORY MAP FOR ARCHIE

By Lee Calcraft

There are some programmers who say that they cannot really get the feel of a machine without understanding its memory map. For those and others, here are some brief notes on the Archimedes' memory map.

The first problem encountered when mapping the memory of an Archimedes is that, unlike the trusty Model B, RAM is not permanently located at a given address. On the model B, you could be sure that there was physical RAM from address &0 to &7FFF. On the Archimedes, the user has no direct access to physical RAM whatsoever. All RAM accessed by the user, whether through indirection operators or from machine code, appears at a given so-called *logical* address. This logical address can range from &0 to &1FFFFFF (0 to 32 Mbytes), and since the machine can have a maximum of 1 Mbyte of physical RAM (4 Mbytes for the 400 series), there will be a great deal of *logical RAM* which does not map directly on to *physical RAM*. By contrast, *every* byte of physical RAM always has a corresponding (though changeable) logical address. If from Basic you try to access RAM at a logical address which is not mapped on to physical RAM, you get the fatal (and untrappable) error "Abort on data transfer". For example try (with no harm to your machine) PRINT ?&A12345.

The accompanying table gives the allocation of RAM on Archimedes machines, and should be valid for both the 300 and 400 series. The precise allocation of RAM within this map is made at power-up, and depends on the configured settings. You are referred to last month's article "Configuring Archimedes" for further details; but to give an example, if you use *CONFIGURE SPRITESIZE 10 you will allocate 10 pages of 8K on a 300 series (or 10 pages of 32K on a 400 series) to sprite use, making 80K (or 320K) in all. This RAM would be allocated from &1400000 upwards. All other user allocations on the table are made in the same way (i.e. upwards from the base address), except for screen RAM. This is allocated *downwards* from the base address (i.e. from &1FFFFFF downwards). To confuse things a little further, the base address of any given screen is the *lowest* memory location of the allocated area (e.g. &1FD8000 on a machine with 160K allocated to screen use). This means that the screen base address (as distinct from the screen allocation base address) is a variable quantity.

You may have noticed one apparent omission from the RAM allocation table. As you can see, there is no individual allocation for font use, even though font space is assigned using the *CONFIGURE command. The reason for this is that font space is taken from the relocatable module area. This means that the amount of RAM reserved in the RMA area is actually the sum of *three* components: the block of RAM automatically taken by the operating system for workspace for resident modules, plus the amount of configured font space, plus the configured RMA space. This can add up to a very large allocation indeed, given that the modules supplied with the series one system require over 110K of workspace.

BASE ADDR	FUNCTION	ALLOCATION	ALTER WITH
1FFF FFF	Screen Memory	0-480K	*CON.SCREENSIZE
1F00 000	Cursor/System Space	32K fixed	
1C00 000	System Heap/Stack	16K-3M	*CON.SYSTEMSIZE
1800 000	Relocatable Modules	0-4M	*CON.RMASIZE
1400 000	Sprite Area	0-4M	*CON.SPRITESIZE
1000 000	RAM Filing System	0-4M	*CON.RAMFSSIZE
0008 000	Applications (eg Basic)	Dynamic	
0000 000	System Space	32K fixed	

As a tailpiece to this article, we shall be publishing next month a program to map out the precise allocation of RAM in any machine over the full 32 Mbyte range. RU

HINTS & TIPS HINTS & TIPS

Another crop of hints, tips and information rounded up by Lee Calcraft.

BUILT IN SCREEN DUMPS

With the new operating system there is a resident module called *Hardcopy,* which provides 3 mono printer dumps for the Epson FX, MX and RX printers. The commands are called *HARDCOPYFX, *HARDCOPYMX and *HARDCOPYRX respectively. The syntax is the same for each:

```
    *HARDCOPY vert/horiz Xscale Yscale
     Margin Threshold
```

The first parameter determines whether the dump is printed normally (0) or widthways (1). The next two give the horizontal and vertical scale, and should normally be set to 1. Larger values will give a larger printout. Next come the left hand margin, and the threshold (0-15) at which the dump output changes from black to white. A good range of parameters to try is:

```
    *HARDCOPYFX 0 1 1 0 5
```

*COUNT FOR COUNTING DISC SPACE

The command *FREE will tell you how much free space you have left on a disc. But with the series one operating system, there is a new and powerful command, *COUNT, for giving much more detailed information. If you type:

```
    *COUNT name
```

it will display the length of the file *name.* But if *name* is a directory it will give the disc space used by that directory, including all sub-directories. The command will accept wildcards, and if you type:

```
    *COUNT $
```

it will give the space used for the whole disc, sub-divided into directories. This is very useful when you need to find out how a disc has suddenly become full.

Better yet, there is a series of options (set using *SET Count$Options, similar to those for *COPY) which allow you to determine recursion, verbose display etc (type *HELP COUNT for details). If you use:

```
    *COUNT $ V
```

you will get a display (or printout) of every file on your disc together with its length, all grouped by directory.

CLEARING THE MOUSE

Although there is an FX call to flush the mouse's input buffer (See Hints and Tips, Vol.1 Issue 2), you can still get problems of one mouse button input overflowing into the next. One way to avoid this is to insert a line which waits until the user has stopped pressing any of the mouse buttons, before the next input is used. A suitable line might take the form:

```
    REPEAT MOUSE X,Y,Z:UNTIL Z=0
```

Depending upon the application, you may need to follow this with a line which waits until a new button is pressed:

```
    REPEAT MOUSE X,Y,Z:UNTIL Z>0
```

ALTERED RUN AND FILE PATHS

On operating systems 0.3 and above, the syntax for the *SET File$Path and *SET Run$Path commands has been altered. On all operating systems later than 0.2, a dot "." must appear as terminator for each file path in the sequence supplied. To take an example, the command:

```
    *SET RUN$PATH ,%,$.MODULES
```

tells operating system 0.2 to look in the current directory, then in the library directory and finally in a directory called $.MODULES, whenever *filename (or *RUN filename) is issued. The space before the first comma is essential.

On all operating systems greater than 0.2 the *SET command must be altered to:

```
    *SET RUN$PATH ,%.,$.MODULES.
```

The same syntax change has occurred with *SET File$Path. This latter command determines the directories searched during other 'read' file operations, including *LOAD.

TWIN AUTO-LOAD

If you use the following alias:

```
    *SET Alias$@LoadType_FFE TWIN %*0
```

every EXEC or Command file (or any other file of type FFE) which you attempt to load will automatically cause TWIN to be loaded in with the selected file in its buffer. This can be very useful when used in conjunction with the RISC User Disc Menu. Whenever you double click with the *adjust* button (i.e. invoking a load) on a Command or EXEC file, it will go into TWIN ready for you to edit it.

The Reference Manual, page 216, states that the syntax of this command has changed on operating systems 0.4 (sic) and above. This does not appear to be the case.

BASIC APPEND

APPEND is a very useful addition to BBC Basic. If you type:

```
    APPEND "filename"
```

the file called filename will be appended to any currently resident Basic program. One particularly useful feature of the command is that the appended file is automatically renumbered to match the numbering of the resident program. You can therefore save procedures and so on as ordinary Basic files, without paying any attention to line numbers. Then whenever they are required, they can be appended to a program under development.

WHICH OPERATING SYSTEM?

There is no simple way to test which operating system you have from within a running program. But the following function does the trick:

```
 10 REM >ostest4
 20 PRINTFNversion
 30 END
 40 :
 50 DEFFNversion
 60 SYS 6+2^17,0,0 TO A%
 70 A$="":A%+=11
 80 FOR B=0 TO 3
 90 A$+=CHR$(A%?B)
100 NEXT:=VAL(A$)
```

The number returned is the operating system number. Early operating systems will return 0.2 or 0.3, while the new series one operating system returns 1.2.

FINDING UNPLUGGED MODULES

You can use ***UNPLUG modulename* to permanently unplug any non-essential relocatable module. It is useful to know that *UNPLUG with no parameter lists all those modules that have been unplugged. Use ***RMREINIT modulename,* followed by Ctrl-Break, to reinstate them. Thanks to Dave Clare for this hint.

DESKTOP FUNCTION KEY RETRIEVAL

After you have used the Desktop, even from the new operating system, you will find that the function keys have been inadvertently left disabled. To retrieve them, use *FX225,1

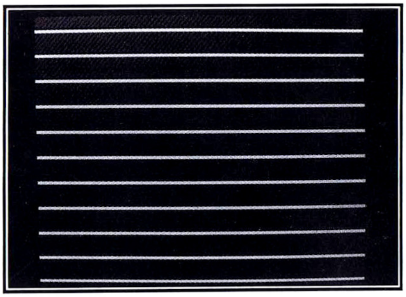
RU

Please send your Hints & Tips to the Editors at the editorial address given at the end of the magazine. All contributions welcomed.

ARCHIMEDES VISUALS (continued from page 17)

BLUE BARS

Our last routine this month generates a static 3D background in shades of blue, which could be used with the quad procedure above to extremely good effect. A set of blue bars are given a 3D effect by the use of a small dark band and a wider bright band between each bar. This background works very well indeed with certain kinds of text display, and has been used in the second of the two programs accompanying the article on anti-alias fonts elsewhere in this issue.



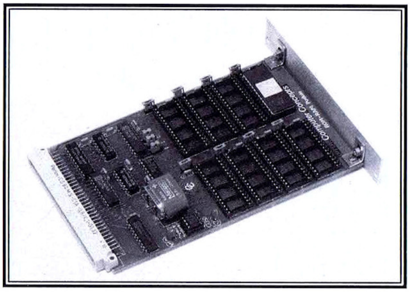
```
 10 REM >BlueBars4
 20 REM 3D Background Effect
 30 REM By Lee Calcraft
 40 :
 50 MODE12
 60 COLOUR9,0,112,208
 70 COLOUR10,0,160,240
 80 COLOUR11,0,0,0
 90 :
100 GCOL128+9:CLG
110 FOR Y=0 TO 1023 STEP 96
120 GCOL10:RECTANGLE FILL 0,Y,1279,12
130 GCOL11:RECTANGLE FILL 0,Y+12,1279,
4
140 NEXT
```

RU

Computer Concepts' ROM Podule

Previewed by Lee Calcraft

The CC ROM Podule should be available by the time you read this brief preview. It works very much like a BBC micro ROM board, and will take 7 ROMs, EPROMs or RAM chips, each of which may be up to 128K in size. It has optional battery back-up, and can give the user a total of 896K of extra storage. The neatly designed board fits inside the Archimedes, and must be plugged into the so-called 'Podule Backplane'. This latter device, supplied by Acorn at around £40, plugs into the Archimedes PCB to create two podule sockets, one of which is used by CC's board.



After installation, and a very simple initialising process, the ROM Podule becomes operational. You can check its presence with the command *ROMPOD n where n is the podule number (normally 0 or 1), depending on whether the board is in the upper or lower podule socket). This displays a list of the ROM and RAM chips plugged into the board. To make use of the podule, you must first engage the ROM filing system (RFS) by typing *RFS. Once you have done this, all filing system star commands are directed to the ROM/RAM filing system implemented on the board rather than to the ADFS (or ANFS). For example, typing *CAT will catalogue all resident ROM and RAM files, while *FREE will give the number of free bytes of RFS RAM, and so on. All the file operations from Basic also apply to the RFS, so to save a file, just type SAVE "filename".

As you may have gathered from the foregoing, the Archimedes does not allow software to be run *directly* from a ROM Podule in the way that the machine's ROM-based modules run from ROM. Each RFS file must be loaded into the computer's main memory first. This loading process is about twice as fast as using the ADFS with floppies, but of course it uses up valuable RAM in the same way as loading files from disc.

Even so, the RFS implemented on CC's ROM podule offers the user a number of advantages. First of all, the speed of loading and the fact that all of its files are permanently resident in the machine (including those in RAM, providing that the optional battery back-up is fitted) makes it useful for storing frequently used utilities and applications. For example, you might keep the 6502 Emulator in podule RAM together with frequently used fonts, and perhaps the Twin editor. Additionally you can store auto-boot files on the RFS, and configure your machine to automatically run these at power up. If you wish, you can blow all these files on to EPROM with the aid of a special WIMPs driven ROM generator program supplied with the podule (though you will also need an EPROM blower for this).

Apart from its massive provision of battery-backed RAM, a major reason for buying a ROM Podule is that all of CC's new software for the Archimedes will only be supplied on ROM, and will only run from a ROM podule. From first impressions, it looks as if their software will be well worth the outlay - especially since they are intending to discount the Podule to purchasers of their Archimedes software, though not the Inter series of products. RU

Computer Concepts ROM Podule, £56.35 inc VAT (battery back-up £11.50 extra).

Computer Concepts
Gaddesden Place, Hemel Hempstead HP2 6EX.
Tel. (0442) 63933

RISC USER magazine

MEMBERSHIP

RISC User is available only on subscription, with a special introductory rate until early 1988. Full subscribers to RISC User may also take out a reduced rate subscription to BEEBUG (the magazine for the BBC micro and Master series).

All subscriptions, including overseas, should be in pounds sterling. We will also accept payment by Connect, Access and Visa, and official UK orders are welcome.

Subscription Rates (12 Months)

	RISC User	**BEEBUG** (reduced rates)
UK, BFPO, Ch.Is	£12.50	£ 8.00
Rest of Europe &Eire	£18.00	£10.00
Middle East	£22.50	£11.00
Americas & Africa	£25.00	£11.50
Elsewhere	£27.00	£12.00

BACK ISSUES

We intend to maintain stocks of back issues New subscribers can therefore obtain earlier copies to provide a complete set from Vol.1 Issue 1. Back issues cost £1.20 each. You should also include postage as shown:

Destination	**UK, BFPO, Ch.Is**	**Europe plus Eire**	**Elsewhere**
First Issue	40p	75p	£2
Each subsequent Issue	20p	45p	85p

MAGAZINE DISC

The programs from each issue of RISC User are available on a monthly 3.5" disc. This will be available to order, or you may take out a subscription to ensure that the disc arrives at the same time as the magazine. The first issue (with six programs and animated graphics demo) is at the special low price of £3.75. The disc for each issue contains all the programs from the magazine, together with a number of additional items by way of demonstration, all at the standard rate of £4.75.

The prices for magazine discs are shown below:

	UK	**Overseas**
Single issue discs	£ 4.75	£ 4.75
Six months subscription	£25.50	£30.00
Twelve months subscription	£50.00	£56.00

Disc subscriptions include postage, but you should add 50p per disc for individual orders.

All orders, subscriptions and other correspondence should be addressed to:

RISC User, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX.
Telephone: St Albans (0727) 40303
(24hrs answerphone service for payment by Connect, Access or Visa card)